

# Machine learning for graphs and with graphs

Graph kernels

Titouan Vayer & Pierre Borgnat

email: [titouan.vayer@inria.fr](mailto:titouan.vayer@inria.fr), [pierre.borgnat@ens-lyon.fr](mailto:pierre.borgnat@ens-lyon.fr)

October 11, 2023



ENS DE LYON

# Table of contents

---

## Kernels in Machine Learning

- A bit of kernels theory

- Back to machine learning: the representer theorem

## Kernels for structured data

- Basics of graphs-kernels

- Focus on Weisfeler-Lehman Kernel

- Conclusion

# Acknowledgments

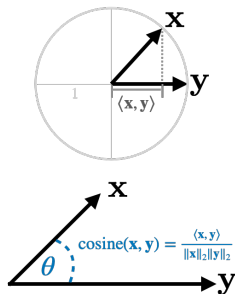
---

Some slides adapted from those of Jean-Philippe Vert and Rémi Flamary.

# What is a kernel ?

## Measuring similarities between objects

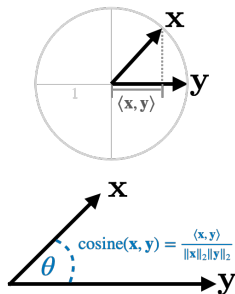
- ▶ Two “objects”  $\mathbf{x}, \mathbf{y}$  in an **abstract space**  $\mathcal{X}$ .
- ▶ A kernel aims at measuring “how similar” is  $\mathbf{x}$  from  $\mathbf{y}$ .
- ▶ e.g.  $\mathcal{X} = \mathbb{R}^d$ ,  $\text{kernel}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$  or cosine similarity.



# What is a kernel ?

## Measuring similarities between objects

- ▶ Two “objects”  $\mathbf{x}, \mathbf{y}$  in an **abstract space**  $\mathcal{X}$ .
- ▶ A kernel aims at measuring “how similar” is  $\mathbf{x}$  from  $\mathbf{y}$ .
- ▶ e.g.  $\mathcal{X} = \mathbb{R}^d$ ,  $\text{kernel}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$  or cosine similarity.



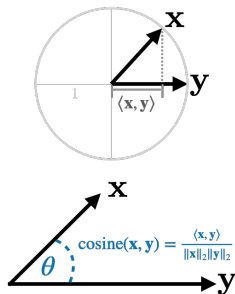
## ML with kernels

- ▶ ML methods based on **pairwise comparisons**.
- ▶ By imposing constraints on the kernel (positive definite), we obtain a **general framework for learning from data** (RKHS).
- ▶ + **without making any assumptions regarding the type of data** (vectors, strings, graphs, images, ...)

# What is a kernel ?

## Measuring similarities between objects

- ▶ Two “objects”  $\mathbf{x}, \mathbf{y}$  in an **abstract space**  $\mathcal{X}$ .
- ▶ A kernel aims at measuring “how similar” is  $\mathbf{x}$  from  $\mathbf{y}$ .
- ▶ e.g.  $\mathcal{X} = \mathbb{R}^d$ ,  $\text{kernel}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$  or cosine similarity.



## ML with kernels

- ▶ ML methods based on **pairwise comparisons**.
- ▶ By imposing constraints on the kernel (positive definite), we obtain a **general framework for learning from data** (RKHS).
- ▶ + **without making any assumptions regarding the type of data** (vectors, strings, graphs, images, ...)

## A principle method for ERM

$\min_{f \in ?} \frac{1}{n} \sum_{i=1}^n \ell(\mathbf{y}_i, f(\mathbf{x}_i)) \rightarrow$  look for  $f$  in specific space (RKHS)

# Table of contents

---

## Kernels in Machine Learning

- A bit of kernels theory

- Back to machine learning: the representer theorem

## Kernels for structured data

- Basics of graphs-kernels

- Focus on Weisfeler-Lehman Kernel

- Conclusion

# The definition

---

## Positive definite (PD) kernel

Let  $\mathcal{X}$  be some space. A function  $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$  is a PD kernel if

- ▶ It is symmetric  $\kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{y}, \mathbf{x})$ .
- ▶ For any  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$  and  $c_1, \dots, c_n \in \mathbb{R}$

$$\sum_{i,j=1}^n c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \quad (1)$$



# The definition

---

## Positive definite (PD) kernel

Let  $\mathcal{X}$  be some space. A function  $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$  is a PD kernel if

- ▶ It is symmetric  $\kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{y}, \mathbf{x})$ .
- ▶ For any  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$  and  $c_1, \dots, c_n \in \mathbb{R}$

$$\sum_{i,j=1}^n c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0. \quad (1)$$

## Remarks

- ▶ (1) equiv.  $\mathbf{K} := (\kappa(\mathbf{x}_i, \mathbf{x}_j))_{ij} \in \mathbb{R}^{n \times n}$  is a PSD matrix  $\forall \mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$ .
- ▶ For  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$  if  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top$  then  $\mathbf{c}^\top \mathbf{K} \mathbf{c} = \|\mathbf{X}^\top \mathbf{c}\|_2^2 \geq 0$ .
- ▶ Works also for  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$  for any  $\Phi$ .
- ▶ Not entirely obvious  $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|_2^2 / 2\sigma^2)$ . (see TD)

# Properties of PD kernel

---

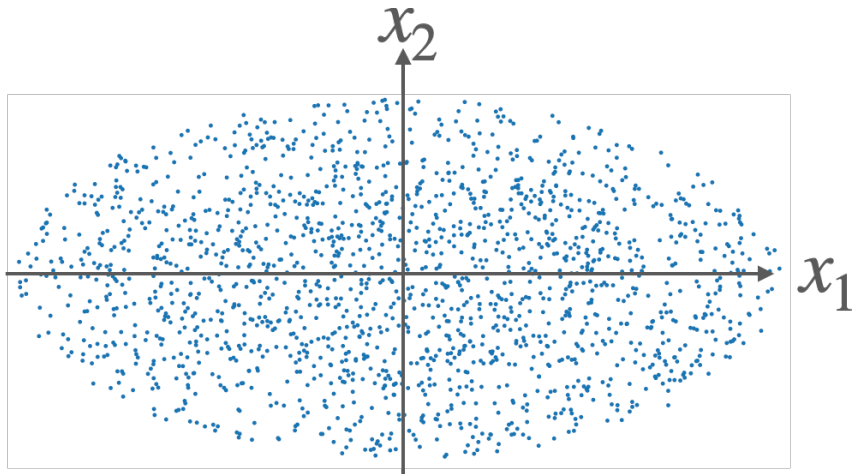
## Basic properties (see TD)

Let  $\kappa_1, \kappa_2, \dots$  be fixed PD kernels.

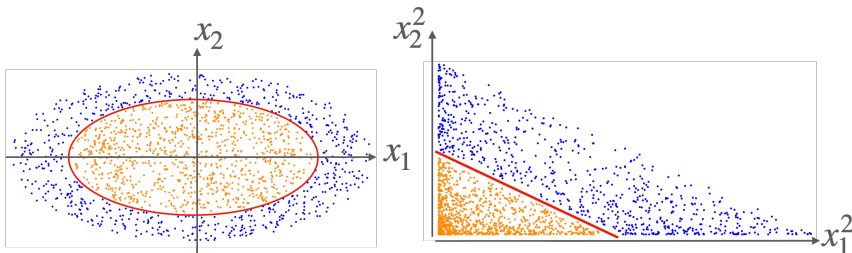
- ▶  $\gamma\kappa_1$  for any  $\gamma > 0$  is a PD kernel.
- ▶  $\kappa_1 + \kappa_2$  is a PD kernel.
- ▶  $\kappa(\mathbf{x}, \mathbf{y}) := \lim_{n \rightarrow +\infty} \kappa_n(\mathbf{x}, \mathbf{y})$  is a PD kernel (provided it exists).
- ▶  $\kappa(\mathbf{x}, \mathbf{y}) := \kappa_1(\mathbf{x}, \mathbf{y})\kappa_2(\mathbf{x}, \mathbf{y})$  is a PD kernel.
- ▶ If  $f : \mathcal{X} \rightarrow \mathbb{R}$  then  $\kappa(\mathbf{x}, \mathbf{y}) := f(\mathbf{x})\kappa_1(\mathbf{x}, \mathbf{y})f(\mathbf{y})$  is a PD kernel.

# Changing the features

---



# Changing the features



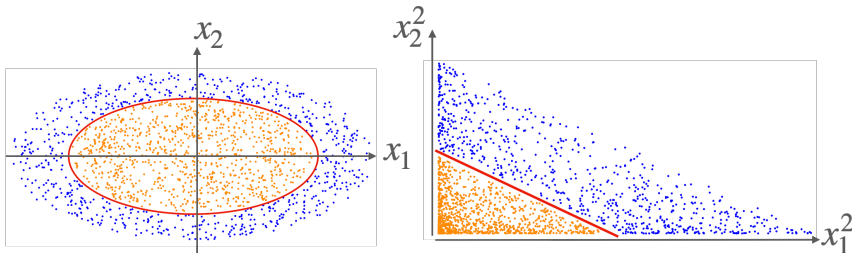
## Polynomial kernel

Consider  $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$  defined by  $\Phi(\mathbf{x} = (x_1, x_2)) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$ . Then:

$$\kappa(\mathbf{x}, \mathbf{y}) := \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathbb{R}^3} = \dots = (\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^2})^2.$$

Basic properties show that it defines a PD kernel.

# Changing the features



## Polynomial kernel

Consider  $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$  defined by  $\Phi(\mathbf{x} = (x_1, x_2)) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$ . Then:

$$\kappa(\mathbf{x}, \mathbf{y}) := \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathbb{R}^3} = \dots = (\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^2})^2.$$

Basic properties show that it defines a PD kernel.

- More generally  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^m$ .

# Translation invariant kernels

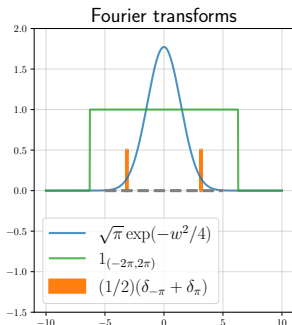
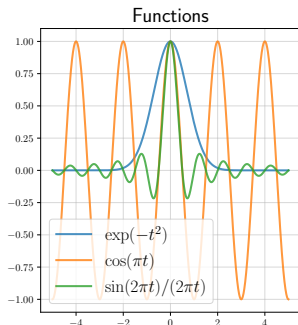
A generic form of kernel on  $\mathcal{X} = \mathbb{R}^d$

- For  $\kappa_0 : \mathbb{R}^d \rightarrow \mathbb{R}$ , kernel defined by

$$\kappa(\mathbf{x}, \mathbf{y}) = \kappa_0(\mathbf{x} - \mathbf{y})$$

- e.g. Gaussian kernel  $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|_2^2 / (2\sigma^2))$ .
- Recall Fourier transform:  $\hat{f}(\boldsymbol{\omega}) = \int_{\mathbb{R}^d} f(\mathbf{x}) e^{-i\langle \boldsymbol{\omega}, \mathbf{x} \rangle} d\mathbf{x}$ .
- Based on Bochner's theorem (see [Wendland 2004, Theorem 6.11](#)):

$$\kappa \text{ is a PD kernel} \iff \forall \boldsymbol{\omega} \in \mathbb{R}^d, \hat{\kappa}_0(\boldsymbol{\omega}) \geq 0$$



# Main property of PD kernel

---

## Main property: Moore–Aronszajn theorem Aronszajn 1950

A function  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a PD kernel if and only if **there exists a Hilbert space  $\mathcal{H}$  and a mapping  $\Phi : \mathcal{X} \rightarrow \mathcal{H}$  such that**

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}, \kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}.$$

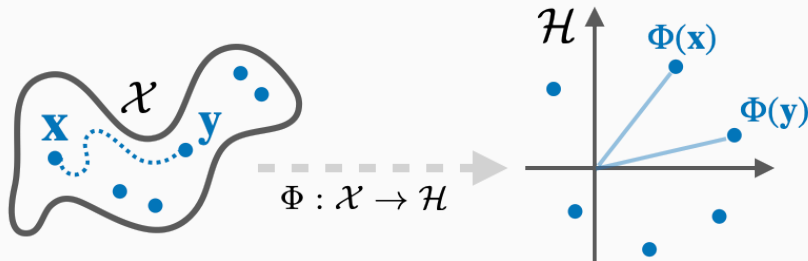
# Main property of PD kernel

## Main property: Moore–Aronszajn theorem Aronszajn 1950

A function  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a PD kernel if and only if **there exists a Hilbert space  $\mathcal{H}$  and a mapping  $\Phi : \mathcal{X} \rightarrow \mathcal{H}$**  such that

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}, \kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}.$$

**Embedding property:**  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}$





# Main property of PD kernel

---

## Some reminders

- ▶  $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{R}$  is a bilinear, symmetric and such that  $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{H}} > 0$  for any  $\mathbf{x} \neq 0$ .
- ▶ A vector space endowed with an inner product is called pre-Hilbert. It is endowed with  $\|\mathbf{x}\|_{\mathcal{H}} := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{H}}}$ .
- ▶ A Hilbert space is a pre-Hilbert space complete for the norm defined by the inner product.

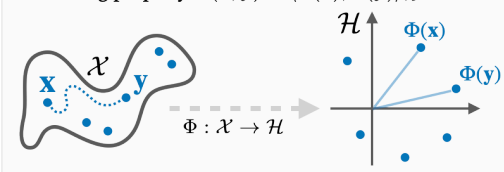
## Proof of the theorem in the discrete case

### On the board

Complete proof [Steinwart and Christmann 2008, Theorem 4.16](#).

# About the feature space

Embedding property:  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}$



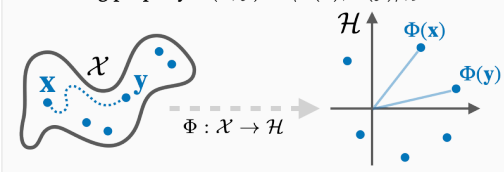
## The feature map $\Phi$ and feature space $\mathcal{H}$

- ▶ The feature space may have **infinite dimension** and is **not unique**.
- ▶ Polynomial kernel in 2D  $\kappa(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle)^2$ :

$$\Phi(\mathbf{x} = (x_1, x_2)) = (x_1^2, x_2^2, x_1 x_2, x_1 x_2), \quad \mathcal{H} = \mathbb{R}^4$$

# About the feature space

Embedding property:  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}$



## The feature map $\Phi$ and feature space $\mathcal{H}$

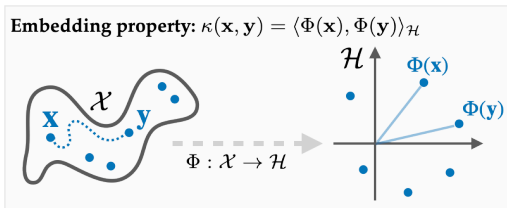
- ▶ The feature space may have **infinite dimension** and is **not unique**.
- ▶ Polynomial kernel in 2D  $\kappa(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle)^2$ :

$$\Phi(\mathbf{x} = (x_1, x_2)) = (x_1^2, x_2^2, x_1 x_2, x_1 x_2), \quad \mathcal{H} = \mathbb{R}^4$$

- ▶ Another possibility:

$$\Phi(\mathbf{x} = (x_1, x_2)) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2), \quad \mathcal{H} = \mathbb{R}^3$$

# About the feature space

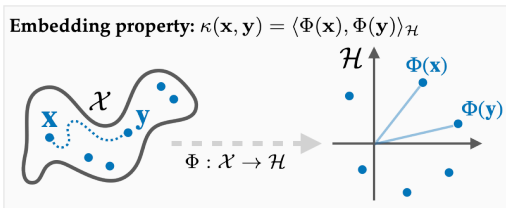


## The feature map $\Phi$ and feature space $\mathcal{H}$

- ▶ The feature space may have **infinite dimension** and is **not unique**.
- ▶ Gaussian Kernel in 1D  $\kappa(x, y) = \exp(-|x - y|_2^2 / (2\sigma^2))$ :

$$\Phi(x) = e^{-\frac{x^2}{2\sigma^2}} \left( 1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!\sigma^4}}x^2, \sqrt{\frac{1}{3!\sigma^6}}x^3, \dots \right)^\top \quad (\text{Taylor series})$$

# About the feature space



## The feature map $\Phi$ and feature space $\mathcal{H}$

- ▶ The feature space may have **infinite dimension** and is **not unique**.
- ▶ Gaussian Kernel in 1D  $\kappa(x, y) = \exp(-|x - y|^2 / (2\sigma^2))$ :

$$\Phi(x) = e^{-\frac{x^2}{2\sigma^2}} \left( 1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!\sigma^4}}x^2, \sqrt{\frac{1}{3!\sigma^6}}x^3, \dots \right)^\top \quad (\text{Taylor series})$$

- ▶ Or  $\mathcal{H} = L_2(\mathbb{R})$  using  $\kappa(x, y) = \frac{1}{\sigma} \sqrt{\frac{2}{\pi}} \int_{-\infty}^{+\infty} \exp(-\frac{(x-t)^2}{\sigma^2}) \exp(-\frac{(y-t)^2}{\sigma^2}) dt$ :

$$\Phi(x) = t \rightarrow \frac{2^{\frac{1}{4}}}{\sqrt{\sigma\pi}^{\frac{1}{4}}} \exp(-\frac{(x-t)^2}{\sigma^2})$$

# Reproducing Kernel Hilbert Space (RKHS)

## From kernels to functions: motivating example

- Kernels can be used to define functions from  $\mathcal{X}$  to  $\mathbb{R}$ .

$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 = \mathcal{H}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \Phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix} \text{ and } f(\mathbf{x}) = a \cdot x_1 + b \cdot x_2 + c \cdot x_1 x_2 \quad (\mathbb{R}^2 \rightarrow \mathbb{R})$$

- Consider  $\boldsymbol{\theta} = (a, b, c)^\top \in \mathcal{H}$  then  $f(\mathbf{x}) = \langle \boldsymbol{\theta}, \Phi(\mathbf{x}) \rangle_{\mathcal{H}}$ .
- **Evaluation of  $f$  at  $\mathbf{x}$  is an inner product in feature space.**

# Reproducing Kernel Hilbert Space (RKHS)

## From kernels to functions: motivating example

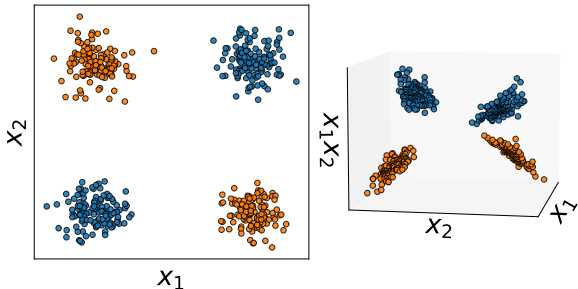
- Kernels can be used to define functions from  $\mathcal{X}$  to  $\mathbb{R}$ .

$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 = \mathcal{H}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \Phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix} \text{ and } f(\mathbf{x}) = a \cdot x_1 + b \cdot x_2 + c \cdot x_1 x_2 \quad (\mathbb{R}^2 \rightarrow \mathbb{R})$$

- Consider  $\theta = (a, b, c)^\top \in \mathcal{H}$  then  $f(\mathbf{x}) = \langle \theta, \Phi(\mathbf{x}) \rangle_{\mathcal{H}}$ .
- **Evaluation of  $f$  at  $\mathbf{x}$  is an inner product in feature space.**

Go into higher dimensions to  
**linearly** separate the classes !



# Reproducing Kernel Hilbert Space (RKHS)

---

## From kernels to functions: first idea

- ▶ **Given**  $\mathcal{H}$  and  $\Phi : \mathcal{X} \rightarrow \mathcal{H}_0$ : defines a kernel  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}_0}$
- ▶ And a space of functions from  $\mathcal{X}$  to  $\mathbb{R}$ .

$$\mathcal{H} := \{f : \exists \boldsymbol{\theta} \in \mathcal{H}_0, \forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \langle \boldsymbol{\theta}, \Phi(\mathbf{x}) \rangle_{\mathcal{H}_0}\}.$$

- ▶ Endowed with the norm

$$\|f\|_{\mathcal{H}} := \inf\{\|\boldsymbol{\theta}\|_{\mathcal{H}_0} : \boldsymbol{\theta} \in \mathcal{H}_0 \text{ with } f = \langle \boldsymbol{\theta}, \Phi(\cdot) \rangle_{\mathcal{H}_0}\} \quad (2)$$

- ▶ It is a Hilbert space of functions called the RKHS of  $\kappa$ .
- ▶ We can stop here... but...



# Reproducing Kernel Hilbert Space (RKHS)

## From kernels to functions: first idea

- ▶ **Given  $\mathcal{H}$  and  $\Phi : \mathcal{X} \rightarrow \mathcal{H}_0$ :** defines a kernel  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}_0}$
- ▶ And a space of functions from  $\mathcal{X}$  to  $\mathbb{R}$ .

$$\mathcal{H} := \{f : \exists \theta \in \mathcal{H}_0, \forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \langle \theta, \Phi(\mathbf{x}) \rangle_{\mathcal{H}_0}\}.$$

- ▶ Endowed with the norm

$$\|f\|_{\mathcal{H}} := \inf\{\|\theta\|_{\mathcal{H}_0} : \theta \in \mathcal{H}_0 \text{ with } f = \langle \theta, \Phi(\cdot) \rangle_{\mathcal{H}_0}\} \quad (2)$$

- ▶ It is a Hilbert space of functions called the RKHS of  $\kappa$ .
- ▶ We can stop here... but...

## From kernels to functions: second idea

- ▶ **Given a PSD kernel  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ .**
- ▶ 1° Find a “suitable”  $(\Phi, \mathcal{H})$  such that  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}$  (recall: many possible)
- ▶ 2° Build upon it to define a suitable space of functions.

# Reproducing Kernel Hilbert Space (RKHS)

## From kernels to functions: first idea

- ▶ **Given**  $\mathcal{H}$  and  $\Phi : \mathcal{X} \rightarrow \mathcal{H}_0$ : defines a kernel  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}_0}$
- ▶ And a space of functions from  $\mathcal{X}$  to  $\mathbb{R}$ .

$$\mathcal{H} := \{f : \exists \theta \in \mathcal{H}_0, \forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \langle \theta, \Phi(\mathbf{x}) \rangle_{\mathcal{H}_0}\}.$$

- ▶ Endowed with the norm

$$\|f\|_{\mathcal{H}} := \inf\{\|\theta\|_{\mathcal{H}_0} : \theta \in \mathcal{H}_0 \text{ with } f = \langle \theta, \Phi(\cdot) \rangle_{\mathcal{H}_0}\} \quad (2)$$

- ▶ It is a Hilbert space of functions called the RKHS of  $\kappa$ .
- ▶ We can stop here... but...

## From kernels to functions: second idea

- ▶ **Given a PSD kernel**  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ .
- ▶ 1° Find a “suitable”  $(\Phi, \mathcal{H})$  such that  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}$  (recall: many possible)
- ▶ 2° Build upon it to define a suitable space of functions. (**RKHS**).

# Reproducing Kernel Hilbert Space (RKHS)

---

Let  $\kappa$  be fixed

- ▶ Among all  $(\Phi, \mathcal{H})$  mentioned in Aronszjan's theorem one  $\mathcal{H}$ , called **RKHS**, is of interest to us.
- ▶ This is a **space of functions from  $\mathcal{X}$  to  $\mathbb{R}$** .
- ▶ Each data point  $\mathbf{x} \in \mathcal{X}$  will be represented **by a function** given by the **canonical feature map**

$$\Phi(\mathbf{x}) = \kappa(\cdot, \mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}$$

# Reproducing Kernel Hilbert Space (RKHS)

---

Let  $\kappa$  be fixed

- ▶ Among all  $(\Phi, \mathcal{H})$  mentioned in Aronszjan's theorem one  $\mathcal{H}$ , called **RKHS**, is of interest to us.
- ▶ This is a **space of functions from  $\mathcal{X}$  to  $\mathbb{R}$** .
- ▶ Each data point  $\mathbf{x} \in \mathcal{X}$  will be represented **by a function** given by the **canonical feature map**

$$\Phi(\mathbf{x}) = \kappa(\cdot, \mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}$$

## Example

- ▶ Consider  $\mathcal{X} = \mathbb{R}$  we could decide to represent  $x \in \mathbb{R}$  as a Gaussian function centered at  $x$ :

$$\Phi(x) = y \rightarrow \exp(-(x - y)^2 / (2\sigma^2))$$

- ▶ What is the corresponding space  $\mathcal{H}$  (if it exists)? What would be the inner-product?

# Reproducing Kernel Hilbert Space (RKHS)

## Reproducing kernel and RKHS

Let  $\mathcal{H}$  be a **Hilbert space** of functions from  $\mathcal{X}$  to  $\mathbb{R}$  with inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ .  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is called a **reproducing kernel** of  $\mathcal{H}$  if

- ▶  $\forall \mathbf{x} \in \mathcal{X}, \kappa(\cdot, \mathbf{x}) \in \mathcal{H}$
- ▶  $\kappa$  satisfies the reproducing property: for any  $f \in \mathcal{H}$ ,

$$\forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}.$$

If a reproducing kernel of  $\mathcal{H}$  exists, then  $\mathcal{H}$  is called a **RKHS**.

# Reproducing Kernel Hilbert Space (RKHS)

## Reproducing kernel and RKHS

Let  $\mathcal{H}$  be a **Hilbert space** of functions from  $\mathcal{X}$  to  $\mathbb{R}$  with inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ .  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is called a **reproducing kernel** of  $\mathcal{H}$  if

- ▶  $\forall \mathbf{x} \in \mathcal{X}, \kappa(\cdot, \mathbf{x}) \in \mathcal{H}$
- ▶  $\kappa$  satisfies the reproducing property: for any  $f \in \mathcal{H}$ ,

$$\forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}.$$

If a reproducing kernel of  $\mathcal{H}$  exists, then  $\mathcal{H}$  is called a **RKHS**.

## Important properties

- ▶ If  $\mathcal{H}$  is a RKHS, then it has a unique reproducing kernel  $\kappa$ .
- ▶ (the feature map is not unique only the kernel is)
- ▶ A function  $\kappa$  can be the reproducing kernel of at most one RKHS.

# Reproducing Kernel Hilbert Space (RKHS)

## Reproducing kernel and RKHS

Let  $\mathcal{H}$  be a **Hilbert space** of functions from  $\mathcal{X}$  to  $\mathbb{R}$  with inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ .  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is called a **reproducing kernel** of  $\mathcal{H}$  if

- ▶  $\forall \mathbf{x} \in \mathcal{X}, \kappa(\cdot, \mathbf{x}) \in \mathcal{H}$
- ▶  $\kappa$  satisfies the reproducing property: for any  $f \in \mathcal{H}$ ,

$$\forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}.$$

If a reproducing kernel of  $\mathcal{H}$  exists, then  $\mathcal{H}$  is called a **RKHS**.

## RKHS and feature spaces

Let  $\mathcal{H}$  be a RKHS with reproducing kernel  $\kappa$ . Then  $\mathcal{H}$  is **one** feature space associated to  $\kappa$ , where the feature map is  $\forall \mathbf{x} \in \mathcal{X}, \Phi(\mathbf{x}) = \kappa(\cdot, \mathbf{x})$ .

# Examples of RKHS

---

So far these functions are a little bit abstract:

## Two questions

- ▶ Given a PD kernel  $\kappa$  what is the RKHS associated to  $\kappa$  ?
- ▶ Given a function space, is it a RKHS and what is the reproducing kernel ?



# Examples of RKHS

---

So far these functions are a little bit abstract:

## Two questions

- ▶ Given a PD kernel  $\kappa$  what is the RKHS associated to  $\kappa$  ?
- ▶ Given a function space, is it a RKHS and what is the reproducing kernel ?

## Battery of examples

- ▶ (on the board) The RKHS associated to  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$  is

$$\mathcal{H} = \{f_{\boldsymbol{\theta}} = \mathbf{x} \rightarrow \langle \boldsymbol{\theta}, \mathbf{x} \rangle; \boldsymbol{\theta} \in \mathbb{R}^d\}$$

endowed with the dot product  $\langle f_{\boldsymbol{\theta}_1}, f_{\boldsymbol{\theta}_2} \rangle_{\mathcal{H}} := \langle \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \rangle$ .

- ▶ (homework) What is the RKHS associated to  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$  ?
- ▶ The space  $L_2(\mathbb{R}^d)$  is **not** a RKHS.

# Examples of RKHS

---

## Battery of examples

- The Paley-Wiener space (bandwidth limited Fourier transform):

$$\mathcal{F}_\pi := \{f \in L_2(\mathbb{R}) : \text{supp } \hat{f} \in [-\pi, \pi]\}$$

where  $\hat{f}$  is the Fourier transform of  $f$ .

# Examples of RKHS

---

## Battery of examples

- ▶ The Paley-Wiener space (bandwidth limited Fourier transform):

$$\mathcal{F}_\pi := \{f \in L_2(\mathbb{R}) : \text{supp } \hat{f} \in [-\pi, \pi]\}$$

where  $\hat{f}$  is the Fourier transform of  $f$ .

- ▶ Inverse Fourier transform

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \hat{f}(\omega) e^{i\omega t} d\omega = \langle \hat{f}, \omega \rightarrow \frac{e^{-i\omega t}}{\sqrt{2\pi}} \rangle_{L_2([-\pi, \pi])}$$

- ▶ Plancherel-Parseval theorem

$$\forall t \in \mathbb{R}, f(t) = \langle \hat{f}, \omega \rightarrow \frac{e^{-i\omega t}}{\sqrt{2\pi}} \rangle_{L_2([-\pi, \pi])} = \langle f, \frac{\sin(\pi(\cdot - t))}{\pi(\cdot - t)} \rangle_{L_2(\mathbb{R})}$$

- ▶ The kernel  $\kappa(s, t) = \frac{\sin(\pi(s-t))}{\pi(s-t)}$

# Examples of RKHS

## Battery of examples

- ▶ The Paley-Wiener space (bandwidth limited Fourier transform):

$$\mathcal{F}_\pi := \{f \in L_2(\mathbb{R}) : \text{supp } \hat{f} \in [-\pi, \pi]\}$$

where  $\hat{f}$  is the Fourier transform of  $f$ .

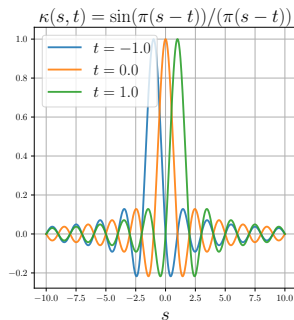
- ▶ Inverse Fourier transform

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \hat{f}(\omega) e^{i\omega t} d\omega = \langle \hat{f}, \omega \rightarrow \frac{e^{-i\omega t}}{\sqrt{2\pi}} \rangle_{L_2([-\pi, \pi])}$$

- ▶ Plancherel-Parseval theorem

$$\forall t \in \mathbb{R}, f(t) = \langle \hat{f}, \omega \rightarrow \frac{e^{-i\omega t}}{\sqrt{2\pi}} \rangle_{L_2([-\pi, \pi])} = \langle f, \frac{\sin(\pi(\cdot - t))}{\pi(\cdot - t)} \rangle_{L_2(\mathbb{R})}$$

- ▶ The kernel  $\kappa(s, t) = \frac{\sin(\pi(s-t))}{\pi(s-t)}$



# Examples of RKHS

---

## Battery of examples

- Translation invariant PD kernels on  $\mathbb{R}^d$   $\kappa(\mathbf{x}, \mathbf{y}) = \kappa_0(\mathbf{x} - \mathbf{y})$  with  $\kappa_0 \in L_1(\mathbb{R}^d) \cap C(\mathbb{R}^d)$  and  $\forall \boldsymbol{\omega} \in \mathbb{R}^d, \hat{\kappa}_0(\boldsymbol{\omega}) \geq 0$ .

# Examples of RKHS

---

## Battery of examples

- ▶ Translation invariant PD kernels on  $\mathbb{R}^d$   $\kappa(\mathbf{x}, \mathbf{y}) = \kappa_0(\mathbf{x} - \mathbf{y})$  with  $\kappa_0 \in L_1(\mathbb{R}^d) \cap C(\mathbb{R}^d)$  and  $\forall \boldsymbol{\omega} \in \mathbb{R}^d, \widehat{\kappa}_0(\boldsymbol{\omega}) \geq 0$ .
- ▶ The corresponding RKHS is

$$\mathcal{H} = \{f \in L_2(\mathbb{R}^d) \cap C(\mathbb{R}^d) : \hat{f}/\sqrt{\widehat{\kappa}_0} \in L_2(\mathbb{R}^d)\}$$

- ▶ The inner product is given by:

$$\langle f, g \rangle_{\mathcal{H}} := (2\pi)^{-d/2} \int_{\mathbb{R}^d} \frac{\hat{f}(\boldsymbol{\omega}) \overline{\hat{g}(\boldsymbol{\omega})}}{\widehat{\kappa}_0(\boldsymbol{\omega})} d\boldsymbol{\omega}.$$

# Examples of RKHS

## Battery of examples

- ▶ Translation invariant PD kernels on  $\mathbb{R}^d$   $\kappa(\mathbf{x}, \mathbf{y}) = \kappa_0(\mathbf{x} - \mathbf{y})$  with  $\kappa_0 \in L_1(\mathbb{R}^d) \cap C(\mathbb{R}^d)$  and  $\forall \boldsymbol{\omega} \in \mathbb{R}^d, \widehat{\kappa}_0(\boldsymbol{\omega}) \geq 0$ .
- ▶ The corresponding RKHS is

$$\mathcal{H} = \{f \in L_2(\mathbb{R}^d) \cap C(\mathbb{R}^d) : \hat{f}/\sqrt{\widehat{\kappa}_0} \in L_2(\mathbb{R}^d)\}$$

- ▶ The inner product is given by:

$$\langle f, g \rangle_{\mathcal{H}} := (2\pi)^{-d/2} \int_{\mathbb{R}^d} \frac{\hat{f}(\boldsymbol{\omega}) \overline{\hat{g}(\boldsymbol{\omega})}}{\widehat{\kappa}_0(\boldsymbol{\omega})} d\boldsymbol{\omega}.$$

- ▶ Special case: Matèrn kernel  $\widehat{\kappa}_0(\boldsymbol{\omega}) \propto (\alpha^2 + \|\boldsymbol{\omega}\|_2^2)^{-s}, s > d/2$
- ▶ Sobolev spaces of order  $s$ :  $\|f\|_{\mathcal{H}}^2 =$  smoothness of the functions as its derivatives in  $L_2(\mathbb{R}^d)$ .

# Reproducing Kernel Hilbert Space (RKHS)

## Reproducing kernels are PD kernels

A function  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a reproducing kernel if and only if it is a PD kernel.



# Reproducing Kernel Hilbert Space (RKHS)

## Reproducing kernels are PD kernels

A function  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a reproducing kernel if and only if it is a PD kernel.

## Remarks

- ▶ One direction easy: a reproducing kernel is a PD kernel (on the board).
- ▶ The other more work: use Moore–Aronszajn theorem +  $\mathcal{F}$  + Steinwart and Christmann 2008, Theorem 4.21.

# Reproducing Kernel Hilbert Space (RKHS)

## Reproducing kernels are PD kernels

A function  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a reproducing kernel if and only if it is a PD kernel.

### Remarks

- ▶ One direction easy: a reproducing kernel is a PD kernel (on the board).
- ▶ The other more work: use Moore–Aronszajn theorem +  $\mathcal{F}$  + Steinwart and Christmann 2008, Theorem 4.21.

### Important consequence

- ▶ Any PSD kernel defines a Hilbert space of functions from  $\mathcal{X}$  to  $\mathbb{R}$ .
- ▶ These functions satisfy

$$\forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}.$$

- ▶ Abstract view of  $\mathcal{H}$ :

$$\mathcal{H} = \overline{\text{Span}\{\kappa(\cdot, \mathbf{x}); \mathbf{x} \in \mathcal{X}\}}.$$

# Table of contents

---

## Kernels in Machine Learning

A bit of kernels theory

Back to machine learning: the representer theorem

## Kernels for structured data

Basics of graphs-kernels

Focus on Weisfeler-Lehman Kernel

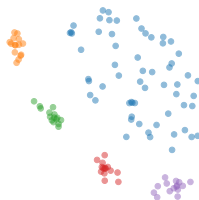
Conclusion

# Recap on supervised ML

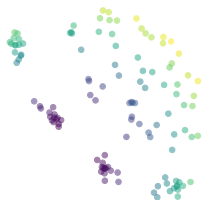
Samples + labels:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Classification



Regression



## Supervised learning

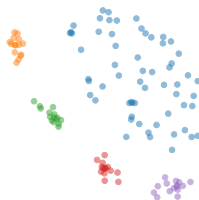
- ▶ The dataset contains the samples  $(\mathbf{x}_i, y_i)_{i=1}^n$  where  $\mathbf{x}_i$  is the feature sample and  $y_i \in \mathcal{Y}$  its label.
- ▶ Prediction space  $\mathcal{Y}$  can be:
  - ▶  $\mathcal{Y} = \{-1, 1\}$  or  $\mathcal{Y} = \{1, \dots, K\}$  for classification problems.
  - ▶  $\mathcal{Y} = \mathbb{R}$  for regression problems ( $\mathbb{R}^p$  for multi-output regression).

# Recap on supervised ML

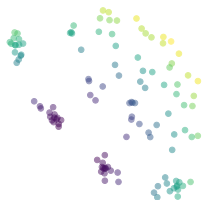
Samples + labels:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Classification



Regression



## Supervised learning

- ▶ The dataset contains the samples  $(\mathbf{x}_i, y_i)_{i=1}^n$  where  $\mathbf{x}_i$  is the feature sample and  $y_i \in \mathcal{Y}$  its label.
- ▶ Prediction space  $\mathcal{Y}$  can be:
  - ▶  $\mathcal{Y} = \{-1, 1\}$  or  $\mathcal{Y} = \{1, \dots, K\}$  for classification problems.
  - ▶  $\mathcal{Y} = \mathbb{R}$  for regression problems ( $\mathbb{R}^p$  for multi-output regression).

## Minimizing the averaged error on the training data

To find  $f : \mathcal{X} \rightarrow \mathcal{Y}$  the idea is to minimize:

$$\min_f \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \text{Reg}(f) \quad (\text{ERM})$$

# Supervised learning

---

## Minimizing the averaged error on the training data

To find  $f : \mathcal{X} \rightarrow \mathcal{Y}$  the idea is to minimize:

$$\min_{f \in ???} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \text{Reg}(f) \quad (\text{ERM})$$

## Problems

- ▶ How to choose the adequate space of functions for  $f$  ?
- ▶ How to properly regularize ?
- ▶ How to efficiently minimize the quantity ?

# Supervised learning

---

## Minimizing the averaged error on the training data

To find  $f : \mathcal{X} \rightarrow \mathcal{Y}$  the idea is to minimize:

$$\min_{f \in ???} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \text{Reg}(f) \quad (\text{ERM})$$

## Problems

- ▶ How to choose the adequate space of functions for  $f$  ?
- ▶ How to properly regularize ?
- ▶ How to efficiently minimize the quantity ?

## One solution

- ▶ When  $\mathcal{Y} \subset \mathbb{R}$  we can consider  $f \in \mathcal{H}$  where  $\mathcal{H}$  is a RKHS.
- ▶ A natural candidate  $\text{Reg}(f) = \|f\|_{\mathcal{H}}^2$ : the higher the smoother  $f$  is.
- ▶ How to ensure that this is not so difficult ?

# Interpretation of minimization on a RKHS

---

- Suppose  $\mathcal{X} = \mathbb{R}^d$  and  $\mathcal{H}$  a RKHS. Consider ERM

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2$$

- Since  $f \in \mathcal{H}$ , then  $f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = \langle f, \Phi(\mathbf{x}) \rangle_{\mathcal{H}}$ .
- Rewriting ERM in RKHS as

$$\min_{\theta \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \langle \theta, \Phi(\mathbf{x}_i) \rangle_{\mathcal{H}}) + \lambda \|\theta\|_{\mathcal{H}}^2$$



# Interpretation of minimization on a RKHS

- ▶ Suppose  $\mathcal{X} = \mathbb{R}^d$  and  $\mathcal{H}$  a RKHS. Consider ERM

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2$$

- ▶ Since  $f \in \mathcal{H}$ , then  $f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = \langle f, \Phi(\mathbf{x}) \rangle_{\mathcal{H}}$ .
- ▶ Rewriting ERM in RKHS as

$$\min_{\theta \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \langle \theta, \Phi(\mathbf{x}_i) \rangle_{\mathcal{H}}) + \lambda \|\theta\|_{\mathcal{H}}^2$$

## Important interpretation

- ▶  $\Phi : \mathcal{X} \rightarrow \mathcal{H}$  pushes the points to a potentially very high-dimensional space (even  $\infty$ ): more powerful representation.
- ▶ Then linear classification/regression is made on this high-dim space  $\mathcal{H}$
- ▶ We can deduce the function in low-dim from the high-dim.

# Interpretation of minimization on a RKHS

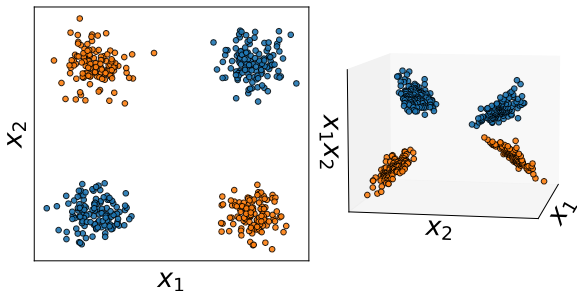
- Suppose  $\mathcal{X} = \mathbb{R}^d$  and  $\mathcal{H}$  a RKHS. Consider ERM

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2$$

- Since  $f \in \mathcal{H}$ , then  $f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = \langle f, \Phi(\mathbf{x}) \rangle_{\mathcal{H}}$ .
- Rewriting ERM in RKHS as

$$\min_{\theta \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \langle \theta, \Phi(\mathbf{x}_i) \rangle_{\mathcal{H}}) + \lambda \|\theta\|_{\mathcal{H}}^2$$

Go into higher dimensions to  
**linearly** separate the classes !



# Interpretation of minimization on a RKHS

- ▶ Suppose  $\mathcal{X} = \mathbb{R}^d$  and  $\mathcal{H}$  a RKHS. Consider ERM

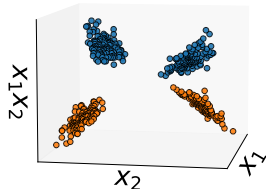
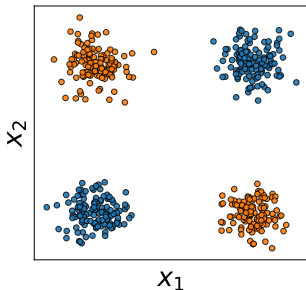
$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2$$

- ▶ Since  $f \in \mathcal{H}$ , then  $f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = \langle f, \Phi(\mathbf{x}) \rangle_{\mathcal{H}}$ .
- ▶ Rewriting ERM in RKHS as

$$\min_{\theta \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \langle \theta, \Phi(\mathbf{x}_i) \rangle_{\mathcal{H}}) + \lambda \|\theta\|_{\mathcal{H}}^2$$

Go into higher dimensions to  
**linearly** separate the classes !

- ▶ But how to implement  $\Phi(\mathbf{x}) \in \mathcal{H}$  on a computer if  $\dim \mathcal{H} = \infty$  ?????
- ▶ How to solve ERM in  $\mathcal{H}$  ????



# The representer theorem

## Main result

- ▶ Let  $\mathcal{X}$  be any space,  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathcal{X}$  a finite set of points.
- ▶  $\mathcal{H}$  a RKHS with reproducing kernel  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ .
- ▶ Let  $\Psi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$  any function that is strictly increasing with respect to the last variable.
- ▶ Then any solution  $f^*$  of the minimization problem

$$\min_{f \in \mathcal{H}} \Psi(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n), \|f\|_{\mathcal{H}}^2)$$

can be written as

$$\forall \mathbf{x} \in \mathcal{X}, f^*(\mathbf{x}) = \sum_{i=1}^n \theta_i \kappa(\mathbf{x}, \mathbf{x}_i) \text{ for some } \boldsymbol{\theta} \in \mathbb{R}^n.$$

# The representer theorem

## Main result

- ▶ Let  $\mathcal{X}$  be any space,  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathcal{X}$  a finite set of points.
- ▶  $\mathcal{H}$  a RKHS with reproducing kernel  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ .
- ▶ Let  $\Psi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$  any function that is strictly increasing with respect to the last variable.
- ▶ Then any solution  $f^*$  of the minimization problem

$$\min_{f \in \mathcal{H}} \Psi(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n), \|f\|_{\mathcal{H}}^2)$$

can be written as

$$\forall \mathbf{x} \in \mathcal{X}, f^*(\mathbf{x}) = \sum_{i=1}^n \theta_i \kappa(\mathbf{x}, \mathbf{x}_i) \text{ for some } \boldsymbol{\theta} \in \mathbb{R}^n.$$

## Important remarks

- ▶ Although the RKHS can be of infinite dimension any solution lives in  $\text{Span}\{\kappa(\cdot, \mathbf{x}_1), \dots, \kappa(\cdot, \mathbf{x}_n)\}$  which is a subspace of dimension  $n$ .
- ▶ Works for any  $\mathcal{X}$  and  $\Psi = \Psi_0 + g$  with  $g \nearrow !!!$

# Practical use of the representer theorem (1/2)

- ▶ When the representer theorem holds we can simply look for  $f$  as

$$\forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \sum_{i=1}^n \theta_i \kappa(\mathbf{x}, \mathbf{x}_i) \text{ for some } \boldsymbol{\theta} \in \mathbb{R}^n.$$

- ▶ Define  $\mathbf{K} := (\kappa(\mathbf{x}_i, \mathbf{x}_j))_{ij}$ .
- ▶ Then , for any  $j \in \llbracket n \rrbracket$

$$f(\mathbf{x}_j) = \sum_{i=1}^n \theta_i \kappa(\mathbf{x}_i, \mathbf{x}_j) = [\mathbf{K}\boldsymbol{\theta}]_j.$$

# Practical use of the representer theorem (1/2)

- ▶ When the representer theorem holds we can simply look for  $f$  as

$$\forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) = \sum_{i=1}^n \theta_i \kappa(\mathbf{x}, \mathbf{x}_i) \text{ for some } \boldsymbol{\theta} \in \mathbb{R}^n.$$

- ▶ Define  $\mathbf{K} := (\kappa(\mathbf{x}_i, \mathbf{x}_j))_{ij}$ .
- ▶ Then , for any  $j \in \llbracket n \rrbracket$

$$f(\mathbf{x}_j) = \sum_{i=1}^n \theta_i \kappa(\mathbf{x}_i, \mathbf{x}_j) = [\mathbf{K}\boldsymbol{\theta}]_j.$$

- ▶ Also

$$\begin{aligned} \|f\|_{\mathcal{H}}^2 &= \left\| \sum_{i=1}^n \theta_i \kappa(\cdot, \mathbf{x}_i) \right\|_{\mathcal{H}}^2 = \left\langle \sum_{i=1}^n \theta_i \kappa(\cdot, \mathbf{x}_i), \sum_{j=1}^n \theta_j \kappa(\cdot, \mathbf{x}_j) \right\rangle_{\mathcal{H}} \\ &= \sum_{ij} \theta_i \theta_j \langle \kappa(\cdot, \mathbf{x}_i), \kappa(\cdot, \mathbf{x}_j) \rangle_{\mathcal{H}} = \sum_{ij} \theta_i \theta_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ &= \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta}. \end{aligned}$$

## Practical use of the representer theorem (2/2)

- Therefore the problem

$$\min_{f \in \mathcal{H}} \Psi(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n), \|f\|_{\mathcal{H}}^2)$$

- is equivalent to

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^n} \Psi([\mathbf{K}\boldsymbol{\theta}]_1, \dots, [\mathbf{K}\boldsymbol{\theta}]_n, \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta})$$

- 1°) To tackle it we only need the Gram matrix  $\mathbf{K}$ : **kernel trick** !
- 2°) Can be used whatever  $\mathcal{X}, \kappa$  !
- 3°) We can solve it on a computer since finite dimensional !
- 4°) It can usually be solved analytically or by numerical methods.



## Practical use of the representer theorem (2/2)

- Therefore the problem

$$\min_{f \in \mathcal{H}} \Psi(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n), \|f\|_{\mathcal{H}}^2)$$

- is equivalent to

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^n} \Psi([\mathbf{K}\boldsymbol{\theta}]_1, \dots, [\mathbf{K}\boldsymbol{\theta}]_n, \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta})$$

- 1°) To tackle it we only need the Gram matrix  $\mathbf{K}$ : **kernel trick** !
- 2°) Can be used whatever  $\mathcal{X}, \kappa$  !
- 3°) We can solve it on a computer since finite dimensional !
- 4°) It can usually be solved analytically or by numerical methods.

### Application to ERM

If we look for  $f$  in a RKHS then we need to solve

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^n \ell(\mathbf{y}_i, [\mathbf{K}\boldsymbol{\theta}]_i) + \lambda \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta}$$

# Application to regression

---

## Setting

- ▶  $\mathbf{x}_i \in \mathcal{X}$  (not necessarily  $\mathbb{R}^d$  !) and  $y_i \in \mathbb{R}$ ,  $\mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$
- ▶ We consider the square loss  $\ell(y, y') = (y - y')^2$
- ▶ The ERM in the RKHS is

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2.$$

# Application to regression

---

## Setting

- ▶  $\mathbf{x}_i \in \mathcal{X}$  (not necessarily  $\mathbb{R}^d$  !) and  $y_i \in \mathbb{R}$ ,  $\mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$
- ▶ We consider the square loss  $\ell(y, y') = (y - y')^2$
- ▶ The ERM in the RKHS is

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2.$$

## Kernel Ridge Regression

The ERM in the RKHS is equivalent to the minimization problem:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^n} \frac{1}{n} \|\mathbf{y} - \mathbf{K}\boldsymbol{\theta}\|_2^2 + \lambda \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta}$$

How can we solve it ? What is the time/memory complexity ?

# Application to regression

---

## Setting

- ▶  $\mathbf{x}_i \in \mathcal{X}$  (not necessarily  $\mathbb{R}^d$  !) and  $y_i \in \mathbb{R}$ ,  $\mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$
- ▶ We consider the square loss  $\ell(y, y') = (y - y')^2$
- ▶ The ERM in the RKHS is

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2.$$

## Kernel Ridge Regression

The ERM in the RKHS is equivalent to the minimization problem:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^n} \frac{1}{n} \|\mathbf{y} - \mathbf{K}\boldsymbol{\theta}\|_2^2 + \lambda \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta}$$

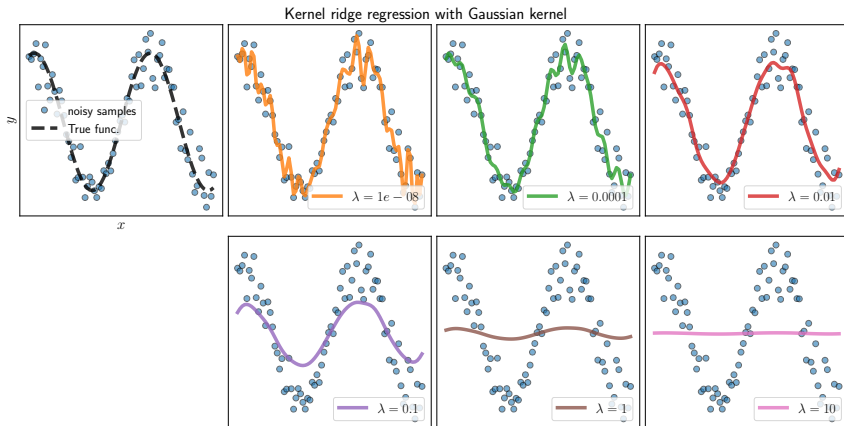
How can we solve it ? What is the time/memory complexity ?

## Solution

Given by  $\boldsymbol{\theta}^* = (\mathbf{K} + \lambda n \mathbf{I})^{-1} \mathbf{y}$ ,  $\forall \mathbf{x} \in \mathcal{X}$ ,  $f^*(\mathbf{x}) = \sum_{i=1}^n \theta_i^* \kappa(\mathbf{x}, \mathbf{x}_i)$ .

# Application to regression

- ▶ Gaussian kernel  $\kappa(x, x') = \exp(-|x - x'|^2 / (2\sigma^2))$
- ▶ Regularization parameter  $\lambda$



# Kernel ridge regression vs linear regression

---

- ▶ Take  $\mathcal{X} = \mathbb{R}^d$  and the linear kernel  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ .
- ▶ Let  $\mathbf{X} = (\mathbf{x}_1, \cdot, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times d}$  the data. The Gram matrix is  $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ .
- ▶ Then corresponding function is

$$f^*(\mathbf{x}) = \sum_{i=1}^n \theta_i^* \kappa(\mathbf{x}, \mathbf{x}_i) = \langle \mathbf{x}, \sum_{i=1}^n \theta_i^* \mathbf{x}_i \rangle = \langle \mathbf{x}, \mathbf{w}^* \rangle.$$

- ▶ We have  $\mathbf{w}^* = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda n \mathbf{I}_n)^{-1} \mathbf{y}$ .

# Kernel ridge regression vs linear regression

- ▶ Take  $\mathcal{X} = \mathbb{R}^d$  and the linear kernel  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ .
- ▶ Let  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times d}$  the data. The Gram matrix is  $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ .
- ▶ Then corresponding function is

$$f^*(\mathbf{x}) = \sum_{i=1}^n \theta_i^* \kappa(\mathbf{x}, \mathbf{x}_i) = \langle \mathbf{x}, \sum_{i=1}^n \theta_i^* \mathbf{x}_i \rangle = \langle \mathbf{x}, \mathbf{w}^* \rangle.$$

- ▶ We have  $\mathbf{w}^* = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda n \mathbf{I}_n)^{-1} \mathbf{y}$ .

$\ell_2$  penalized linear regression: ridge regression

The problem

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2 \text{ has solution } \mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda n \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{y}.$$

# Kernel ridge regression vs linear regression

- ▶ Take  $\mathcal{X} = \mathbb{R}^d$  and the linear kernel  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ .
- ▶ Let  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times d}$  the data. The Gram matrix is  $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ .
- ▶ Then corresponding function is

$$f^*(\mathbf{x}) = \sum_{i=1}^n \theta_i^* \kappa(\mathbf{x}, \mathbf{x}_i) = \langle \mathbf{x}, \sum_{i=1}^n \theta_i^* \mathbf{x}_i \rangle = \langle \mathbf{x}, \mathbf{w}^* \rangle.$$

- ▶ We have  $\mathbf{w}^* = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda n \mathbf{I}_n)^{-1} \mathbf{y}$ .

## $\ell_2$ penalized linear regression: ridge regression

The problem

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2 \text{ has solution } \mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda n \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{y}.$$

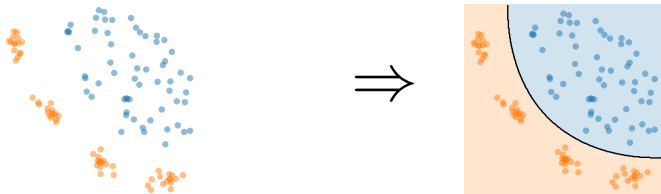
## Matrix inversion lemma

$$(\mathbf{X}^\top \mathbf{X} + \lambda n \mathbf{I}_d)^{-1} \mathbf{X}^\top = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda n \mathbf{I}_n)^{-1}$$

- ▶ Both agree !
- ▶ Complexity roughly: KRR  $O(n^3)$ , RR  $O(\min\{d^3, n^3\})$ .



# Binary classification



## Objective

$$(\mathbf{x}_i, y_i)_{i=1}^n \Rightarrow f : \mathbb{R}^d \rightarrow \{-1, 1\}$$

- ▶ Train a function  $f(\mathbf{x}) = y \in \mathcal{Y}$  predicting a binary value ( $\mathcal{Y} = \{-1, 1\}$ ).
- ▶  $f(\mathbf{x}) = 0$  defines the boundary on the partition of the feature space.

## ERM in RKHS

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2.$$

# Loss functions

---

A focus on classification problems  $\mathcal{Y} = \{-1, 1\}$

$$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i)) \text{ with } \Phi \text{ non-increasing.}$$

# Loss functions

---

A focus on classification problems  $\mathcal{Y} = \{-1, 1\}$

$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i))$  with  $\Phi$  non-increasing.

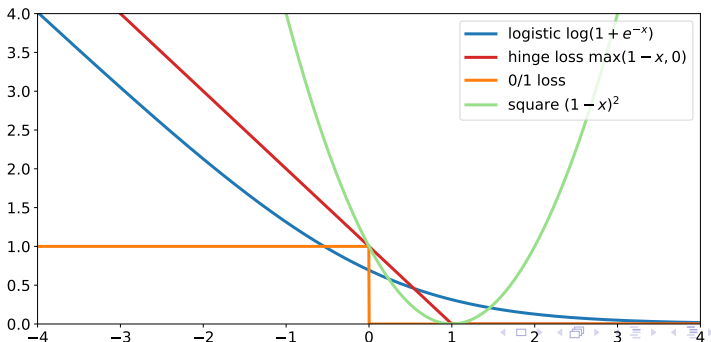
- ▶  $y_i f(\mathbf{x}_i)$  is **the margin** (on the board).
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$  (0/1 loss)
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$  (hinge loss: **SVM**)
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$  (logistic loss)

# Loss functions

A focus on classification problems  $\mathcal{Y} = \{-1, 1\}$

$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i))$  with  $\Phi$  non-increasing.

- ▶  $y_i f(\mathbf{x}_i)$  is **the margin** (on the board).
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$  (0/1 loss)
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$  (hinge loss: **SVM**)
- ▶  $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$  (logistic loss)



# Support Vector Machines (SVM)

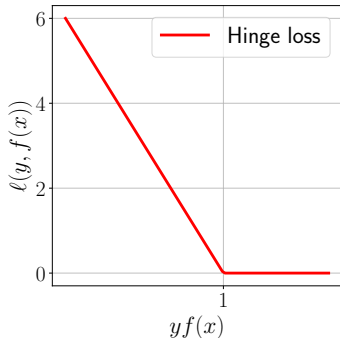
## Definition

- ▶ The hinge-loss is the function  $\mathbb{R} \rightarrow \mathbb{R}_+$ :

$$\begin{aligned}\Phi_{\text{hinge}}(x) &= \max(1 - x, 0) \\ &= \begin{cases} 0 & \text{if } x \geq 1 \\ 1 - x & \text{otherwise} \end{cases}\end{aligned}$$

- ▶ SVM is the corresponding large-margin classifier, which solves:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \Phi_{\text{hinge}}(y_i f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2.$$



# Support Vector Machines (SVM)

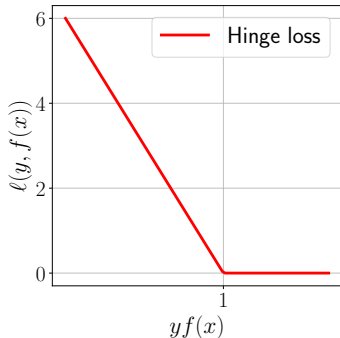
## Definition

- ▶ The hinge-loss is the function  $\mathbb{R} \rightarrow \mathbb{R}_+$ :

$$\begin{aligned}\Phi_{\text{hinge}}(x) &= \max(1 - x, 0) \\ &= \begin{cases} 0 & \text{if } x \geq 1 \\ 1 - x & \text{otherwise} \end{cases}\end{aligned}$$

- ▶ SVM is the corresponding large-margin classifier, which solves:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \Phi_{\text{hinge}}(y_i f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2.$$

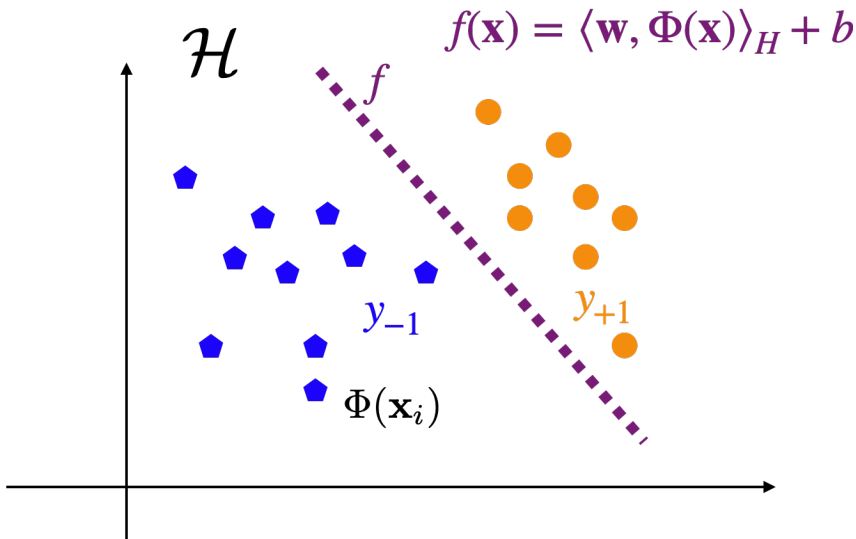


## Solving for the SVM (details in Steinwart and Christmann 2008)

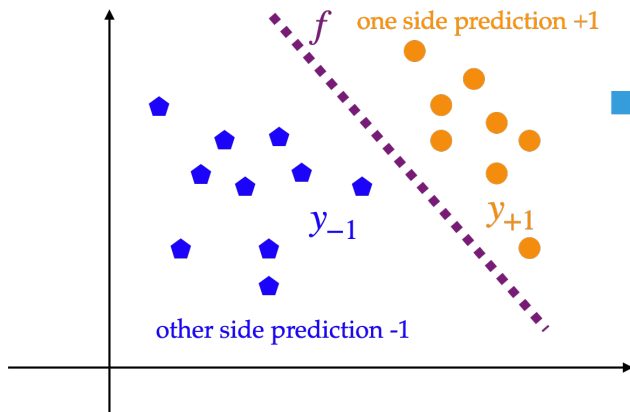
- ▶ Representer theorem: sol. of the form  $f^*(\mathbf{x}) = \sum_{i=1}^n \theta_i^* \kappa(\mathbf{x}, \mathbf{x}_i)$ .
- ▶  $\theta^*$  can be found by solving a quadratic program (QP).
- ▶ Again: we only need to know the Gram matrix  $\mathbf{K} = (\kappa(\mathbf{x}_i, \mathbf{x}_j))_{ij}$ .

# What is SVM doing ?

Find a separating hyperplane in the RKHS



# What is SVM doing ?

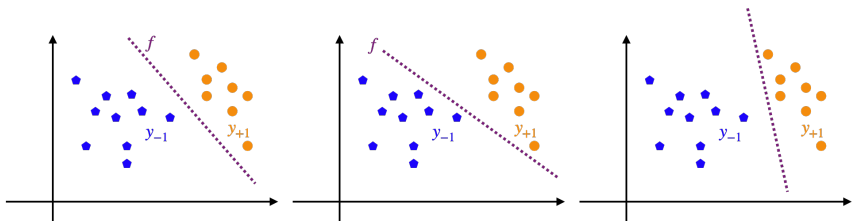


■ Classification:  
 $\text{sign}(f)$   
if  $f(\mathbf{x}) > 0 \rightarrow +1$   
if  $f(\mathbf{x}) < 0 \rightarrow -1$

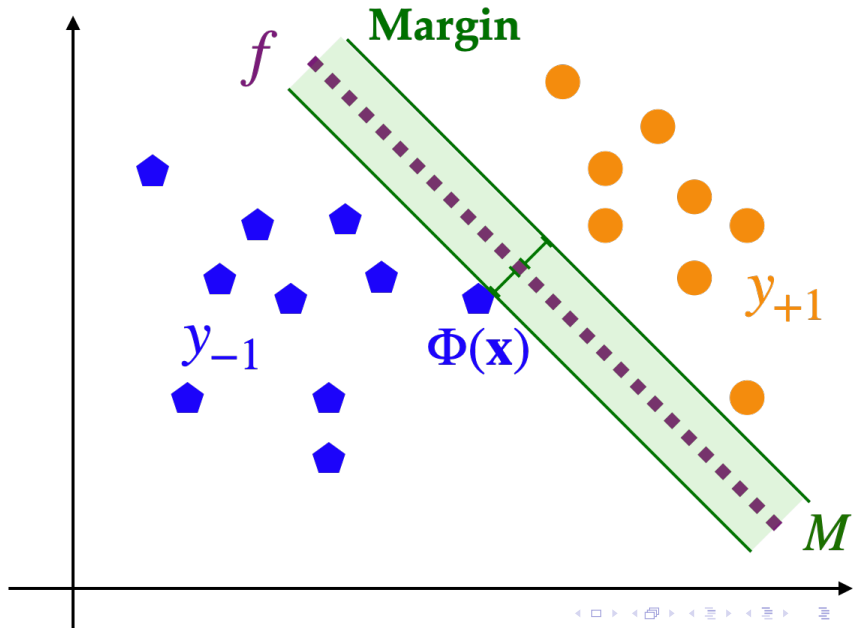


# What is SVM doing ?

But there could be an infinity of separating hyperplanes or zero !

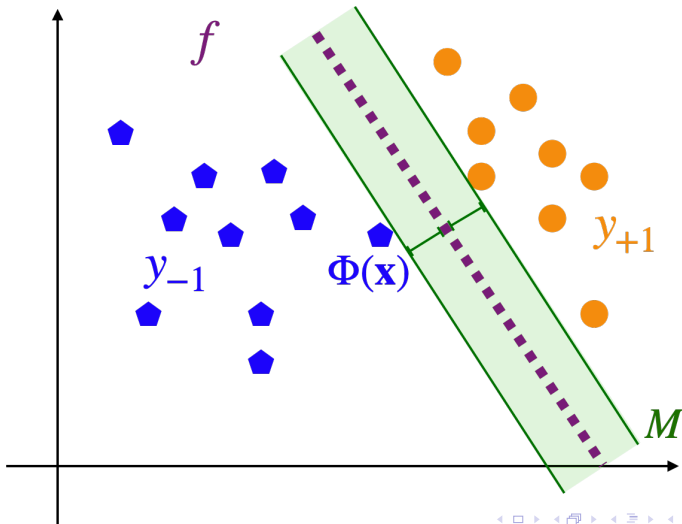


# What is SVM doing ?



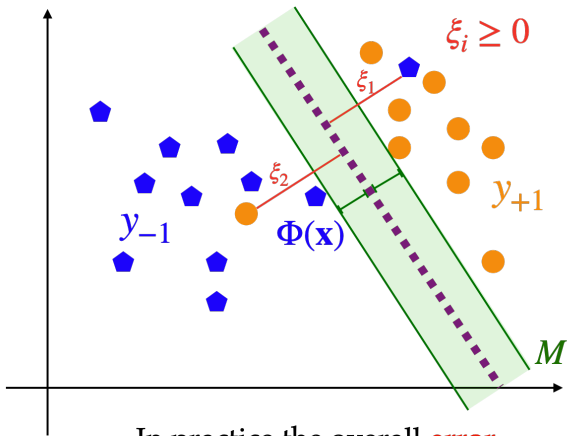
# What is SVM doing ?

SVM finds the hyperplane that maximizes the **margin**



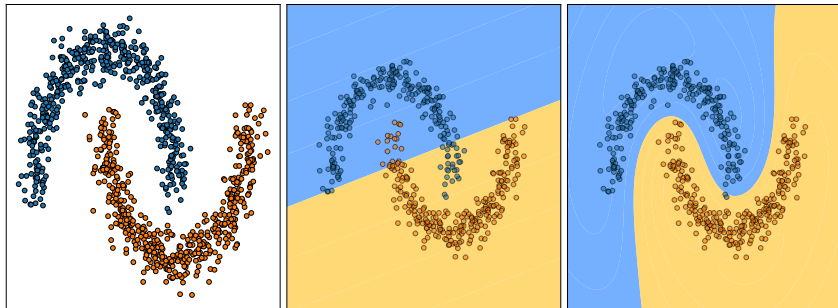
# What is SVM doing ?

+ We allow **some errors** to be made



In practice the overall **error** is controlled by a regularization param.  $C$

# Example



# Conclusion

---

- ▶ Kernel theory is very rich, kernels are quite simple but also versatile.
- ▶ Defines a very general way of learning classifiers/regressors on any kind of space.
- ▶ Based on the representer theorem: we only need the Gram matrix !
- ▶ Difficulties: the choice of the kernel (see TD), also can be expensive.

# Table of contents

---

## Kernels in Machine Learning

- A bit of kernels theory

- Back to machine learning: the representer theorem

## Kernels for structured data

- Basics of graphs-kernels

- Focus on Weisfeler-Lehman Kernel

- Conclusion

# Kernels for structured data

---

## Objective

Given a dataset of graphs  $(G_1, \dots, G_n)$  can we build machine learning models to do:

- ▶ Supervised learning: each graph associated to  $y_i \in \mathcal{Y}$ .
- ▶ Unsupervised learning: PCA, Kernel PCA, graph embedding...



# Kernels for structured data

---

## Objective

Given a dataset of graphs  $(G_1, \dots, G_n)$  can we build machine learning models to do:

- ▶ Supervised learning: each graph associated to  $y_i \in \mathcal{Y}$ .
- ▶ Unsupervised learning: PCA, Kernel PCA, graph embedding...

## Application of RKHS for graphs

Let  $\mathcal{X} = \{ \text{set of all graphs} \}$  can we build interesting kernels

$\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  ?

- ▶ For  $G, G' \in \mathcal{X}$ ,  $\kappa(G, G')$  is a notion of “similarity” between graphs.
- ▶ Gram matrix  $\mathbf{K} = (\kappa(G_i, G_j))_{(i,j) \in \llbracket n \rrbracket^2}$ .
- ▶ Then do stuff...

# Kernels for structured data

---

## Objective

Given a dataset of graphs  $(G_1, \dots, G_n)$  can we build machine learning models to do:

- ▶ Supervised learning: each graph associated to  $y_i \in \mathcal{Y}$ .
- ▶ Unsupervised learning: PCA, Kernel PCA, graph embedding...

## Application of RKHS for graphs

Let  $\mathcal{X} = \{ \text{set of all graphs} \}$  can we build interesting kernels

$\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  ?

- ▶ For  $G, G' \in \mathcal{X}$ ,  $\kappa(G, G')$  is a notion of “similarity” between graphs.
- ▶ Gram matrix  $\mathbf{K} = (\kappa(G_i, G_j))_{(i,j) \in [n]^2}$ .
- ▶ Then do stuff...

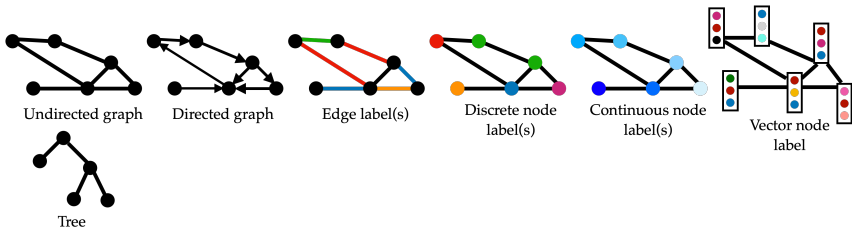
## Some notations

A graph  $G = (V, E)$ . Labeling function if attributes/labels  $\ell_G : V \cup E \rightarrow S$   
( $S$  discrete or continuous  $\subset \mathbb{R}^N$ )

# What is a good graph kernel ?

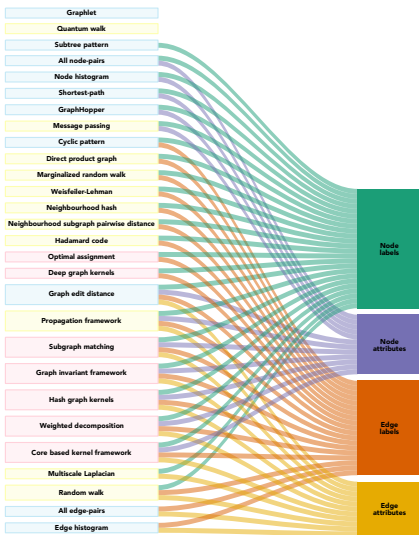
## Properties of the graph kernel

- ▶ Handle graphs that are directed (or undirected) ?
- ▶ Handle node or edge labels or attributes that are present in the graphs?
- ▶ Efficient to compute ? Complexity *w.r.t.*  $|V|, |E|, \dim$  ?
- ▶ Is there a particular relevant substructure (e.g. tree patterns) that would preclude the choice of a particular kernel?



# The kernel jungle

Surveys: K. Borgwardt et al. 2020; Nikolettos, Siglidis, and Vazirgiannis 2021



Graph Kernel	Exp. $\phi$	Node Labels	Node Attributes	Type	Complexity
Vertex Histogram	✓	✓	✗	R-convolution	$\mathcal{O}(n)$
Edge Histogram	✓	✓	✗	R-convolution	$\mathcal{O}(m)$
Random Walk	✗ <sup>†</sup>	✓	✓	R-convolution	$\mathcal{O}(n^3)$
Subtree	✗	✓	✓	R-convolution	$\mathcal{O}(n^{2.4deg} h)$
Cyclic Pattern	✓	✓	✗	intersection	$\mathcal{O}((c+2)n+2m)$
Shortest Path	✗ <sup>†</sup>	✓	✓	R-convolution	$\mathcal{O}(n^4)$
Graphlet	✓	✗	✗	R-convolution	$\mathcal{O}(n^k)$
Weisfeiler-Lehman Subtree	✓	✓	✗	R-convolution	$\mathcal{O}(hm)$
Neighborhood Hash	✓	✓	✗	intersection	$\mathcal{O}(hm)$
Neighborhood Subgraph Pairwise Distance	✓	✓	✗	R-convolution	$\mathcal{O}(n^2 m \log(m))$
Lovász $\vartheta$	✓	✗	✗	R-convolution	$\mathcal{O}(n(s + \frac{m}{n}) + s^2)$
SVM- $\vartheta$	✓	✗	✗	R-convolution	$\mathcal{O}(n(s + n^2) + s^2)$
Ordered Decomposition DAGs	✓	✓	✗	R-convolution	$\mathcal{O}(n \log n)$
Pyramid Match	✗	✓	✗	assignment	$\mathcal{O}(ndL)$
Weisfeiler-Lehman Optimal Assignment	✗	✓	✗	assignment	$\mathcal{O}(hm)$
Subgraph Matching	✗	✓	✓	R-convolution	$\mathcal{O}(kn^{k+1})$
GraphHopper	✗	✓	✓	R-convolution	$\mathcal{O}(n^4)$
Graph Invariant Kernels	✗	✓	✓	R-convolution	$\mathcal{O}(n^6)$
Propagation	✓	✓	✓	R-convolution	$\mathcal{O}(hm)$
Multiscale Laplacian	✗	✓	✓	R-convolution	$\mathcal{O}(n^6 h)$

# Bag of structures

---

A majority of graph kernels are instances of the *convolution kernels* Haussler et al. 1999.

## Principle

- ▶ Compare graphs by first dividing them into substructures of various granularity.
- ▶ E.g. vertices, subgraphs, all shortest paths of a graph.
- ▶ Defining *base kernels* at the fine granularity and combine them.
- ▶ Of the form  $\kappa(G, G') = \sum_{r \in \mathcal{R}, r' \in \mathcal{R}'} \kappa_{\text{substructure}}(r, r')$ .

# Bag of structures

---

A majority of graph kernels are instances of the *convolution kernels* Haussler et al. 1999.

## Principle

- ▶ Compare graphs by first dividing them into substructures of various granularity.
- ▶ E.g. vertices, subgraphs, all shortest paths of a graph.
- ▶ Defining *base kernels* at the fine granularity and combine them.
- ▶ Of the form  $\kappa(G, G') = \sum_{r \in \mathcal{R}, r' \in \mathcal{R}'} \kappa_{\text{substructure}}(r, r')$ .

## Advantages & limitations

- ▶ Intuitive definitions + relatively good results.
- ▶ Sometimes computational limitations.
- ▶ Expressiveness limitations.
- ▶ “Diagonal dominance problem” Yanardag and Vishwanathan 2015.

# All node-pairs kernel

---

## A first idea

- ▶ Given  $G = (V, E)$ ,  $G' = (V', E')$ ,
- ▶ Suppose the labels of the nodes of both graphs are in  $S$ .
- ▶ Consider a kernel on the nodes

$$\kappa_{\text{node}} : S \times S \rightarrow \mathbb{R}$$

- ▶ The all node-pairs kernel is defined by

$$\kappa(G, G') = \sum_{v \in V} \sum_{v' \in V'} \kappa_{\text{node}}(\ell_G(v), \ell_{G'}(v'))$$

# All node-pairs kernel

---

## A first idea

- ▶ Given  $G = (V, E)$ ,  $G' = (V', E')$ ,
- ▶ Suppose the labels of the nodes of both graphs are in  $S$ .
- ▶ Consider a kernel on the nodes

$$\kappa_{\text{node}} : S \times S \rightarrow \mathbb{R}$$

- ▶ The all node-pairs kernel is defined by

$$\kappa(G, G') = \sum_{v \in V} \sum_{v' \in V'} \kappa_{\text{node}}(\ell_G(v), \ell_{G'}(v'))$$

## Remarks

- ▶ Runtime in  $O(|V| \times |V'| \times \dim(S))$ .
- ▶ Can handle discrete/continuous labels.
- ▶ Does not take into account the structures of the graphs.



# Node histogram kernel

## A baseline kernel (1/2)

- Suppose the labels are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \dots, \sigma_{|\Sigma|}\}$$

- The node histogram kernel is defined as

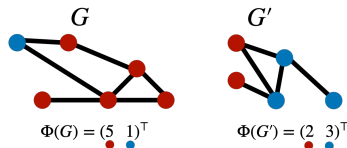
$$\kappa_{\text{NH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle.$$

where

$$\Phi(G) = \left( \sum_{v \in V} \mathbf{1}_{\ell_G(v)=\sigma_1}, \dots, \sum_{v \in V} \mathbf{1}_{\ell_G(v)=\sigma_{|\Sigma|}} \right).$$

- Simply corresponds to an unnormalised histogram that counts the occurrence of each node label in the graph.

Node histogram kernel



# Node histogram kernel

## A baseline kernel (1/2)

- Suppose the labels are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \dots, \sigma_{|\Sigma|}\}$$

- The node histogram kernel is defined as

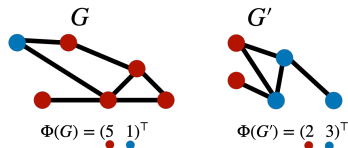
$$\kappa_{\text{NH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle.$$

where

$$\Phi(G) = \left( \sum_{v \in V} \mathbf{1}_{\ell_G(v)=\sigma_1}, \dots, \sum_{v \in V} \mathbf{1}_{\ell_G(v)=\sigma_{|\Sigma|}} \right).$$

- Simply corresponds to an unnormalised histogram that counts the occurrence of each node label in the graph.

Node histogram kernel



## Remarks

- Can be computed in  $O(|V| + |V'|)$ .
- Does not take into account the structures of the graphs.

# Node histogram kernel

## A baseline kernel (1/2)

- Suppose the labels are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \dots, \sigma_{|\Sigma|}\}$$

- The node histogram kernel is defined as

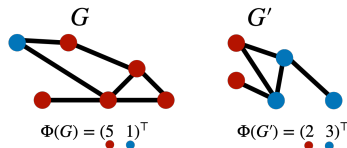
$$\kappa_{\text{NH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle.$$

where

$$\Phi(G) = \left( \sum_{v \in V} \mathbf{1}_{\ell_G(v) = \sigma_1}, \dots, \sum_{v \in V} \mathbf{1}_{\ell_G(v) = \sigma_{|\Sigma|}} \right).$$

- Simply corresponds to an unnormalised histogram that counts the occurrence of each node label in the graph.

Node histogram kernel



## Remarks

- Can be computed in  $O(|V| + |V'|)$ .
- Does not take into account the structures of the graphs.
- Of the form  $\kappa_{\text{NH}}(G, G') = \sum_{v \in V, v' \in V'} \mathbf{1}_{\ell_G(v) = \ell_{G'}(v')}.$

# Edge histogram kernel

## A baseline kernel (2/2)

- Suppose the **edges labels** are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \dots, \sigma_{|\Sigma|}\}$$

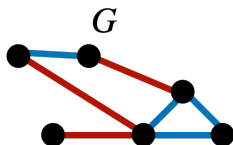
- The edge histogram kernel is defined as

$$\kappa_{\text{EH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle.$$

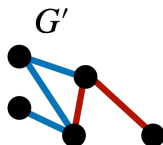
where  $\Phi(G) =$

$$\left( \sum_{e \in E} \mathbf{1}_{\ell(e)=\sigma_1}, \dots, \sum_{e \in E} \mathbf{1}_{\ell(e)=\sigma_{|\Sigma|}} \right).$$

Edge histogram kernel



$$\Phi(G) = \begin{pmatrix} 3 & 4 \end{pmatrix}^T$$



$$\Phi(G') = \begin{pmatrix} 2 & 3 \end{pmatrix}^T$$

# Edge histogram kernel

## A baseline kernel (2/2)

- Suppose the **edges labels** are discrete over a finite alphabet

$$\Sigma = \{\sigma_1, \dots, \sigma_{|\Sigma|}\}$$

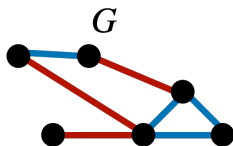
- The edge histogram kernel is defined as

$$\kappa_{\text{EH}}(G, G') = \langle \Phi(G), \Phi(G') \rangle.$$

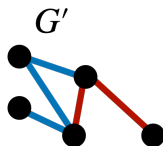
where  $\Phi(G) =$

$$\left( \sum_{e \in E} \mathbf{1}_{\ell(e)=\sigma_1}, \dots, \sum_{e \in E} \mathbf{1}_{\ell(e)=\sigma_{|\Sigma|}} \right).$$

Edge histogram kernel



$$\Phi(G) = \begin{pmatrix} 3 & 4 \end{pmatrix}^T$$



$$\Phi(G') = \begin{pmatrix} 2 & 3 \end{pmatrix}^T$$

## Remarks

- Can be computed in  $O(|E| + |E'|)$ .
- Does not take into account the labels of the nodes.
- Can be combined with the previous one as

$$\kappa(G, G') = \kappa_{\text{EH}}(G, G') \times \kappa_{\text{NH}}(G, G')$$

# The shortest-path kernel

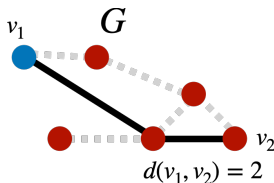
K. M. Borgwardt and Kriegel 2005

- ▶ Compute all pair-to-pair shortest-paths in  $G, G'$  with Floyd-Warshall.
- ▶ The kernel is defined as

$$\kappa_{\text{SP}}(G, G') = \sum_{(v_1, v_2) \in V} \sum_{(v'_1, v'_2) \in V'} \kappa_0(d(v_1, v_2), d(v'_1, v'_2)).$$

where  $d(v_1, v_2)$  is the shortest-path distance between  $v_1, v_2$ .

- ▶  $\kappa_0$  is a kernel that compares the lengths of the two shortest-paths.
- ▶  $\kappa_0(x, y) = x \times y$  (linear kernel) or  $\kappa_0(x, y) = \mathbf{1}_{x=y}$  (dirac).



# The shortest-path kernel

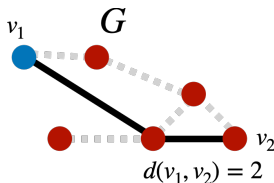
K. M. Borgwardt and Kriegel 2005

- ▶ Compute all pair-to-pair shortest-paths in  $G, G'$  with Floyd-Warshall.
- ▶ The kernel is defined as

$$\kappa_{SP}(G, G') = \sum_{(v_1, v_2) \in V} \sum_{(v'_1, v'_2) \in V'} \kappa_0(d(v_1, v_2), d(v'_1, v'_2)).$$

where  $d(v_1, v_2)$  is the shortest-path distance between  $v_1, v_2$ .

- ▶  $\kappa_0$  is a kernel that compares the lengths of the two shortest-paths.
- ▶  $\kappa_0(x, y) = x \times y$  (linear kernel) or  $\kappa_0(x, y) = \mathbf{1}_{x=y}$  (dirac).



## Remarks

- ▶ Complexity Floyd-Warshall on  $G, O(|V|^3)$ .
- ▶ Variants with Bellman-Ford's, Dijkstra's algorithms.
- ▶ General complexity for  $\kappa_{SP}$   $O(|V|^2|V'|^2)$ .
- ▶ Many variants **with attributes**.

# GraphHopper kernel

---

Undirected graphs with edge weights and node attributes.

- ▶ Even for real-valued/vector attributes [Feragen et al. 2013](#).
- ▶ Kernel is defined as

$$\kappa_{\text{GH}}(G, G') = \sum_{p \in \mathcal{P}_G} \sum_{p' \in \mathcal{P}_{G'}} \kappa_0(p, p') \text{ where } \mathcal{P}_G: \text{ set of **all shortest-paths**.}$$



# GraphHopper kernel

Undirected graphs with edge weights and node attributes.

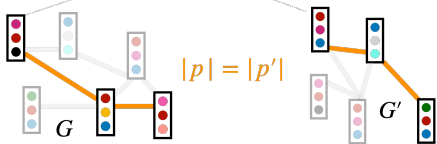
► Even for real-valued/vector attributes [Feragen et al. 2013](#).

► Kernel is defined as

$$\kappa_{\text{GH}}(G, G') = \sum_{p \in \mathcal{P}_G} \sum_{p' \in \mathcal{P}_{G'}} \kappa_0(p, p') \text{ where } \mathcal{P}_G: \text{ set of **all shortest-paths**.}$$

► Base kernel  $\kappa_0(p, p') = \begin{cases} \sum_{j=1}^{|p|} \kappa_{\text{node}}(p_j, p'_j) & \text{if equal length } |p| = |p'| \\ 0 & \text{otherwise} \end{cases}$

$$\kappa_0(p, p') = \kappa_{\text{node}} \left( \begin{bmatrix} \text{pink} \\ \text{red} \\ \text{black} \end{bmatrix}, \begin{bmatrix} \text{pink} \\ \text{blue} \end{bmatrix} \right) + \kappa_{\text{node}} \left( \begin{bmatrix} \text{red} \\ \text{yellow} \\ \text{blue} \end{bmatrix}, \begin{bmatrix} \text{blue} \\ \text{cyan} \end{bmatrix} \right) + \kappa_{\text{node}} \left( \begin{bmatrix} \text{pink} \\ \text{orange} \end{bmatrix}, \begin{bmatrix} \text{green} \\ \text{blue} \end{bmatrix} \right)$$



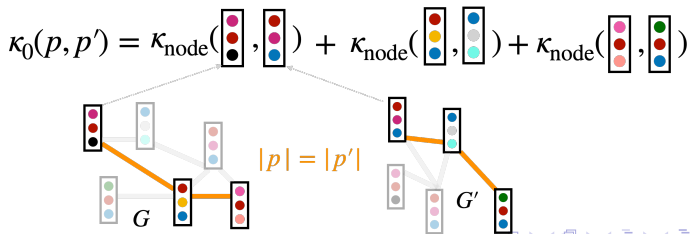
# GraphHopper kernel

Undirected graphs with edge weights and node attributes.

- ▶ Even for real-valued/vector attributes [Feragen et al. 2013](#).
- ▶ Interestingly averaged overall worst-case complexity  $O(|V||V'| \dim(S))$ .
- ▶ Kernel is defined as

$$\kappa_{\text{GH}}(G, G') = \sum_{p \in \mathcal{P}_G} \sum_{p' \in \mathcal{P}_{G'}} \kappa_0(p, p') \text{ where } \mathcal{P}_G: \text{ set of **all shortest-paths**.}$$

- ▶ Base kernel  $\kappa_0(p, p') = \begin{cases} \sum_{j=1}^{|p|} \kappa_{\text{node}}(p_j, p'_j) & \text{if equal length } |p| = |p'| \\ 0 & \text{otherwise} \end{cases}$

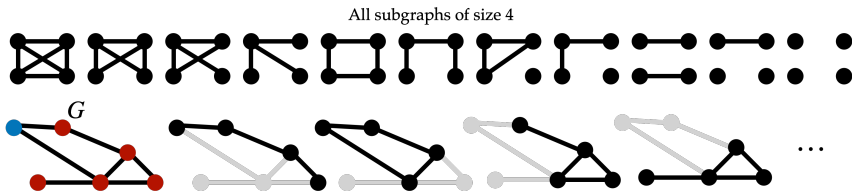


# The Graphlet kernel

Principle Shervashidze, Vishwanathan,  
et al. 2009

- ▶ Count substructures in graphs.
- ▶ Graphlet = subgraph with  $k$  vertices.
- ▶  $\mathbb{G} := \{g_1, \dots, g_{N_k}\}$  set of  $k$ -graphlets (asymptotically  $N_k \approx 2^{\binom{k}{2}}/k!$ ).
- ▶ Kernel  $\kappa(G, G') = \langle \Phi(G), \Phi(G') \rangle$

$$\Phi(G) \propto (|\{g_i \in G\}|, \dots, |\{g_{N_k} \in G\}|)^T$$



Different size 4 graphlets found in  $G$

# The Graphlet kernel

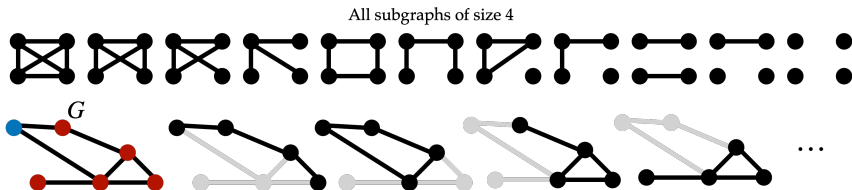
**Principle** Shervashidze, Vishwanathan,  
et al. 2009

- ▶ Count substructures in graphs.
- ▶ Graphlet = subgraph with  $k$  vertices.
- ▶  $\mathbb{G} := \{g_1, \dots, g_{N_k}\}$  set of  $k$ -graphlets (asymptotically  $N_k \approx 2^{\binom{k}{2}}/k!$ ).
- ▶ Kernel  $\kappa(G, G') = \langle \Phi(G), \Phi(G') \rangle$

$$\Phi(G) \propto (|\{g_i \in G\}|, \dots, |\{g_{N_k} \in G\}|)^T$$

## Remarks

- ▶ Ignores all labels.
- ▶ Computational bottleneck: enumeration of all graphlets.
- ▶ Complexity in  $O(|V|^k)$  time.
- ▶ Typically  $k \in \{3, 4, 5\}$ .
- ▶ Counting all possible subgraphs is NP-hard  
Gärtner, Flach, and Wrobel 2003.



Different size 4 graphlets found in  $G$

# The graph isomorphism problem

---

## Checking if two graphs are “identical”

Two graphs  $G = (V, E)$ ,  $G' = (V', E')$  are **isomorphic** ( $G \cong G'$ ) if there exists a **bijection**  $\Psi : V \rightarrow V'$  such that

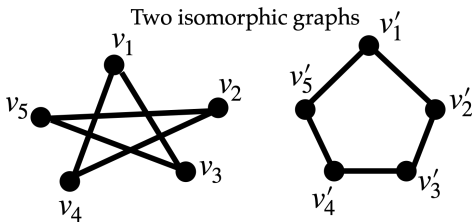
$$(u, v) \in E \iff (\Psi(u), \Psi(v)) \in E'.$$

# The graph isomorphism problem

## Checking if two graphs are “identical”

Two graphs  $G = (V, E)$ ,  $G' = (V', E')$  are **isomorphic** ( $G \cong G'$ ) if there exists a **bijection**  $\Psi : V \rightarrow V'$  such that

$$(u, v) \in E \iff (\Psi(u), \Psi(v)) \in E'.$$



## Remark

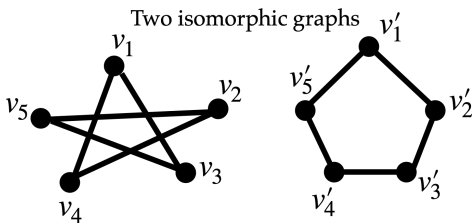
- ▶ Same graphs up to a permutation.
- ▶ Currently no known polynomial-time algorithms for solving this problem.
- ▶ Not known to be NP-complete.
- ▶ Quasi-polynomial algorithm [Babai 2016](#).

# The graph isomorphism problem

## Checking if two graphs are “identical”

Two graphs  $G = (V, E)$ ,  $G' = (V', E')$  are **isomorphic** ( $G \cong G'$ ) if there exists a **bijection**  $\Psi : V \rightarrow V'$  such that

$$(u, v) \in E \iff (\Psi(u), \Psi(v)) \in E'.$$



## Remark

- ▶ Same graphs up to a permutation.
- ▶ Currently no known polynomial-time algorithms for solving this problem.
- ▶ Not known to be NP-complete.
- ▶ Quasi-polynomial algorithm Babai 2016.

Weisfeiler-Lehman test of isomorphism Leman and Weisfeiler 1968

On the board

# Table of contents

---

## Kernels in Machine Learning

- A bit of kernels theory

- Back to machine learning: the representer theorem

## Kernels for structured data

- Basics of graphs-kernels

- Focus on Weisfeler-Lehman Kernel**

- Conclusion



# Weisfeiler–Lehman kernel

---

A very popular graph kernel based on Shervashidze, Schweitzer, et al. 2011

- ▶ Originally handle graphs with discrete labels.
- ▶ Uses **iterative label refinement**.
- ▶ Concepts from the Weisfeiler-Lehman test of isomorphism.

# Weisfeiler–Lehman kernel

---

A very popular graph kernel based on Shervashidze, Schweitzer, et al. 2011

- ▶ Originally handle graphs with discrete labels.
- ▶ Uses **iterative label refinement**.
- ▶ Concepts from the Weisfeiler-Lehman test of isomorphism.

## Graphs relabeling/refinement

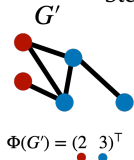
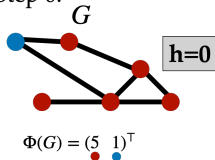
- ▶ Recursively refine the node labels by applying local transformations

$$a_v = \text{AGGREGATE} \left( \{ \{ \ell_G^{(\text{old})}(v'); v' \in \mathcal{N}(v) \} \} \right)$$
$$\text{and } \ell_G^{(\text{new})}(v) = \text{COMBINE} \left( \ell_G^{(\text{old})}(v), a_v \right).$$

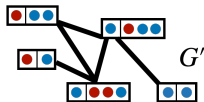
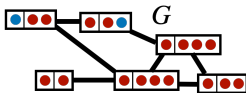
- ▶ This general idea can give rise to a multitude of distinct graph kernels:
- ▶ (i) the specific form of COMBINE, AGGREGATE.
- ▶ (ii) which kernels are used to compare the resulting modified graphs.
- ▶ (iii) how the graph at multiple scales are aggregated into a single value.

# Weisfeiler–Lehman kernel

Step 0:

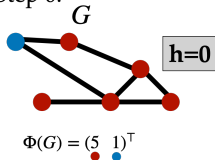


Step 1: « Enrich » the labels with neighbors

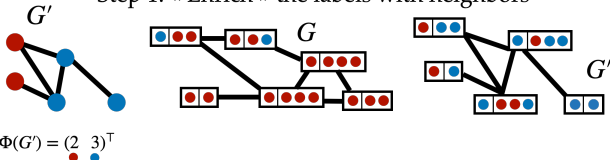


# Weisfeiler–Lehman kernel

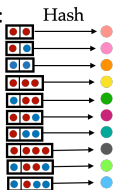
Step 0:



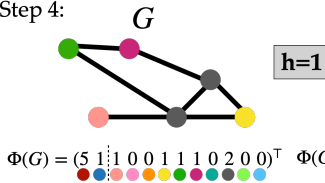
Step 1: « Enrich » the labels with neighbors



Step 3:



Step 4:



-> Step 0 ...  
-> Terminates  
after  
h iterations

# Weisfeiler–Lehman kernel

---

## The Weisfeiler–Lehman kernel

- ▶ The function AGGREGATE sorts in alphabetic order.
- ▶ The function COMBINE hashes to compress the tuple into a single integer-valued label.
- ▶ Produces a sequence of graphs  $(G_0, \dots, G_h)$ .
- ▶ The Weisfeiler–Lehman kernel is

$$\kappa_{\text{WL}}(G, G') = \sum_{i=0}^h \kappa_0(G_i, G'_i),$$

for a base kernel  $\kappa_0$ .

# Weisfeiler–Lehman kernel

---

## The Weisfeiler–Lehman kernel

- ▶ The function AGGREGATE sorts in alphabetic order.
- ▶ The function COMBINE hashes to compress the tuple into a single integer-valued label.
- ▶ Produces a sequence of graphs  $(G_0, \dots, G_h)$ .
- ▶ The Weisfeiler–Lehman kernel is

$$\kappa_{\text{WL}}(G, G') = \sum_{i=0}^h \kappa_0(G_i, G'_i),$$

for a base kernel  $\kappa_0$ .

- ▶ Most common  $\kappa_0$  *subtree kernel*:  $\Phi(G)$  = number of occurrences of each label in the alphabet of all compressed labels at each step.
- ▶ Complexity: for one graph  $O(|E| \times h)$ .
- ▶ Runtime scales only linearly with the number of edges !

# Optimal assignment kernel

---

## General setting (Kriege, Giscard, and Wilson 2016)

- ▶ Different than “bag of structure” kernels.
- ▶ Let  $X, Y \subset \Omega$  with  $|X| = |Y|$ .

$$\kappa_{OA}(X, Y) = \max_{B \in \mathcal{B}(X, Y)} \sum_{x \in X} \kappa_0(x, B(y)) \text{ where } \mathcal{B}(X, Y) = \text{all bijections.}$$

- ▶  $\kappa$  is a valid PSD kernel if  $\kappa_0 : \Omega \times \Omega \rightarrow \mathbb{R}_+$  is *strong*:

$$\kappa_0(x, y) \geq \min\{\kappa_0(x, z), \kappa_0(z, y)\} \quad \forall (x, y, z).$$

- ▶ Assign the parts of one objects to the parts of the other *s.t.* the total similarity is maximum possible.

# Optimal assignment kernel

---

## General setting (Kriege, Giscard, and Wilson 2016)

- ▶ Different than “bag of structure” kernels.
- ▶ Let  $X, Y \subset \Omega$  with  $|X| = |Y|$ .

$$\kappa_{OA}(X, Y) = \max_{B \in \mathcal{B}(X, Y)} \sum_{x \in X} \kappa_0(x, B(y)) \text{ where } \mathcal{B}(X, Y) = \text{all bijections.}$$

- ▶  $\kappa$  is a valid PSD kernel if  $\kappa_0 : \Omega \times \Omega \rightarrow \mathbb{R}_+$  is *strong*:

$$\kappa_0(x, y) \geq \min\{\kappa_0(x, z), \kappa_0(z, y)\} \quad \forall (x, y, z).$$

- ▶ Assign the parts of one objects to the parts of the other *s.t.* the total similarity is maximum possible.

## Weisfeiler-Lehman optimal assignment kernel

- ▶  $i \in \llbracket h \rrbracket$ ,  $\tau_i(v)$  denotes the color of vertex  $v$  at step  $i$  of the WL process.
- ▶ The base kernel is  $\kappa_0(v, v') = \sum_{i=0}^h \mathbf{1}_{\tau_i(v)=\tau_i(v')} + \text{padding}$ .
- ▶ Can also be computed in  $O(hm)$ .



# Continuous alternative to Weisfeiler–Lehman

---

## Hash graph kernel Morris et al. 2016

- ▶ Let  $\kappa$  be a graph kernel (such as WL).
- ▶  $\mathfrak{H} = \{\mathfrak{h}_1, \mathfrak{h}_2, \dots\}$  a family of hash functions.
- ▶  $\mathfrak{h}_i : \mathbb{R}^d \rightarrow \mathbb{N}$  is a hash function.
- ▶  $\mathfrak{h}_i(G)$ : the discretised graph resulting from applying  $\mathfrak{h}_i$  to continuous attributes of the graph.
- ▶ The kernel is defined as

$$\kappa_{\text{HGK}}(G, G') = \frac{1}{|\mathfrak{H}|} \sum_{i \in \mathfrak{H}} \kappa(\mathfrak{h}_i(G), \mathfrak{h}_i(G')).$$

# Continuous alternative to Weisfeiler–Lehman

## Hash graph kernel Morris et al. 2016

- ▶ Let  $\kappa$  be a graph kernel (such as WL).
- ▶  $\mathfrak{H} = \{\mathfrak{h}_1, \mathfrak{h}_2 \dots\}$  a family of hash functions.
- ▶  $\mathfrak{h}_i : \mathbb{R}^d \rightarrow \mathbb{N}$  is a hash function.
- ▶  $\mathfrak{h}_i(G)$ : the discretised graph resulting from applying  $\mathfrak{h}_i$  to continuous attributes of the graph.
- ▶ The kernel is defined as

$$\kappa_{\text{HGK}}(G, G') = \frac{1}{|\mathfrak{H}|} \sum_{i \in \mathfrak{H}} \kappa(\mathfrak{h}_i(G), \mathfrak{h}_i(G')).$$

## Example of hash functions

- ▶ Locality-sensitive hashing schemes Datar et al. 2004.
- ▶ Idea: if  $\mathbf{x}, \mathbf{y}$  are “close” then  $\mathbb{P}[\mathfrak{h}_1(\mathbf{x}) = \mathfrak{h}_2(\mathbf{y})]$  is “high” and conversely.
- ▶ More collusion for nearby points.
- ▶ e.g.  $\mathfrak{h}(\mathbf{x}) = \lfloor \frac{\langle \mathbf{x}, \mathbf{a} \rangle + b}{r} \rfloor, \mathbf{a} \sim \mu, b \sim \text{unif}([0, r])$

# Table of contents

---

## Kernels in Machine Learning

- A bit of kernels theory

- Back to machine learning: the representer theorem

## Kernels for structured data

- Basics of graphs-kernels

- Focus on Weisfeler-Lehman Kernel

- Conclusion







# Conclusion

---

- ▶ Graph kernels are very simple but powerful way of using all the ML machinery on graphs.
- ▶ The big question is to choose the “right” kernel.
- ▶ No straight answer, it depends on the task.
- ▶ In practice: always use simple graph kernels as baselines.






# References I

---

-  Aronszajn, Nachman (1950). “Theory of reproducing kernels”. In: *Transactions of the American mathematical society* 68.3, pp. 337–404.
-  Babai, László (2016). “Graph isomorphism in quasipolynomial time”. In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 684–697.
-  Borgwardt, Karsten et al. (2020). “Graph kernels: State-of-the-art and future challenges”. In: *Foundations and Trends® in Machine Learning* 13.5-6, pp. 531–712.
-  Borgwardt, Karsten M and Hans-Peter Kriegel (2005). “Shortest-path kernels on graphs”. In: *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE, 8–pp.
-  Datar, Mayur et al. (2004). “Locality-sensitive hashing scheme based on p-stable distributions”. In: *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 253–262.
-  Feragen, Aasa et al. (2013). “Scalable kernels for graphs with continuous attributes”. In: *Advances in neural information processing systems* 26.

# References II

---

-  Gärtner, Thomas, Peter Flach, and Stefan Wrobel (2003). “On graph kernels: Hardness results and efficient alternatives”. In: *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*. Springer, pp. 129–143.
-  Haussler, David et al. (1999). *Convolution kernels on discrete structures*. Tech. rep. Citeseer.
-  Kriege, Nils M, Pierre-Louis Giscard, and Richard Wilson (2016). “On valid optimal assignment kernels and applications to graph classification”. In: *Advances in neural information processing systems* 29.
-  Leman, AA and Boris Weisfeiler (1968). “A reduction of a graph to a canonical form and an algebra arising during this reduction”. In: *Nauchno-Tekhnicheskaya Informatsiya* 2.9, pp. 12–16.
-  Morris, Christopher et al. (2016). “Faster kernels for graphs with continuous attributes via hashing”. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, pp. 1095–1100.

# References III

---

-  Nikolentzos, Giannis, Giannis Siglidis, and Michalis Vazirgiannis (2021). “Graph kernels: A survey”. In: *Journal of Artificial Intelligence Research* 72, pp. 943–1027.
-  Shervashidze, Nino, Pascal Schweitzer, et al. (2011). “Weisfeiler-lehman graph kernels.”. In: *Journal of Machine Learning Research* 12.9.
-  Shervashidze, Nino, SVN Vishwanathan, et al. (2009). “Efficient graphlet kernels for large graph comparison”. In: *Artificial intelligence and statistics*. PMLR, pp. 488–495.
-  Steinwart, Ingo and Andreas Christmann (2008). *Support vector machines*. Springer Science & Business Media.
-  Wendland, Holger (2004). *Scattered data approximation*. Vol. 17. Cambridge university press.
-  Yanardag, Pinar and SVN Vishwanathan (2015). “Deep graph kernels”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374.