

# Advanced Learning Models

## Part on Kernel Methods

Julien Mairal

`julien.mairal@inria.fr`

Inria, Grenoble



## History of the course



These slides are due in large part to Jean-Philippe Vert, who gave the course on kernel methods from 2004 to 2015 at the master MVA at ENS Cachan.

- Along the years, the course has become more and more exhaustive and the slides are probably one of the best reference available on kernels.

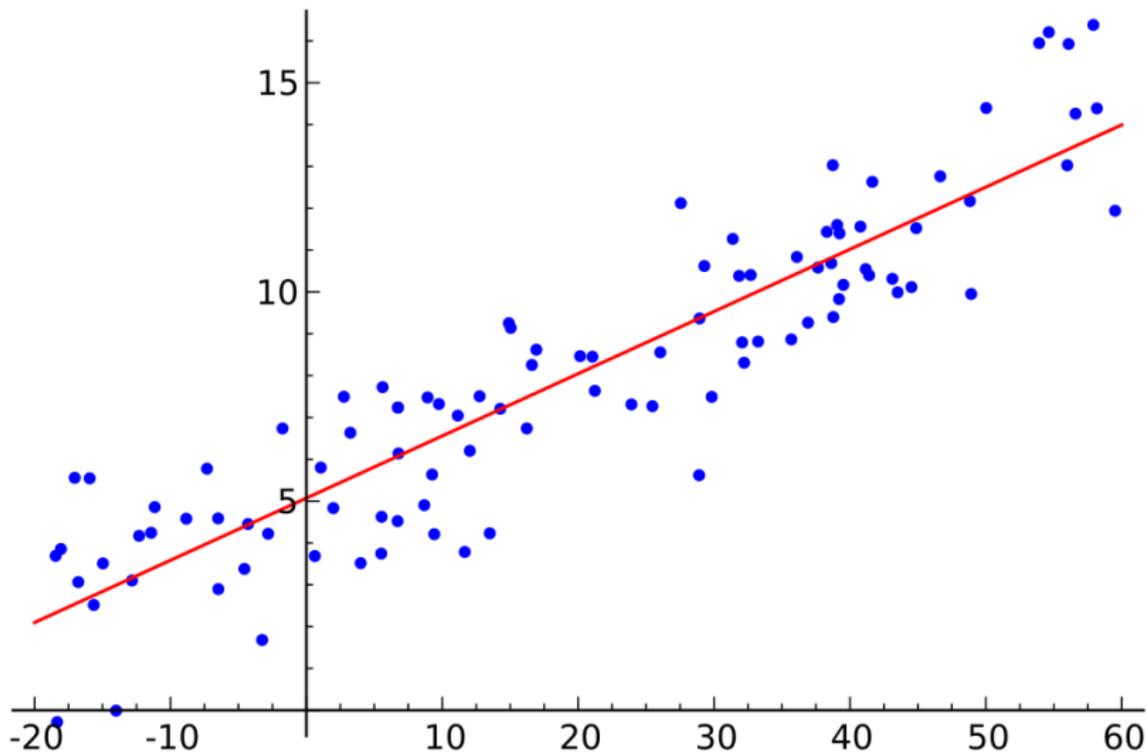
## History of the course



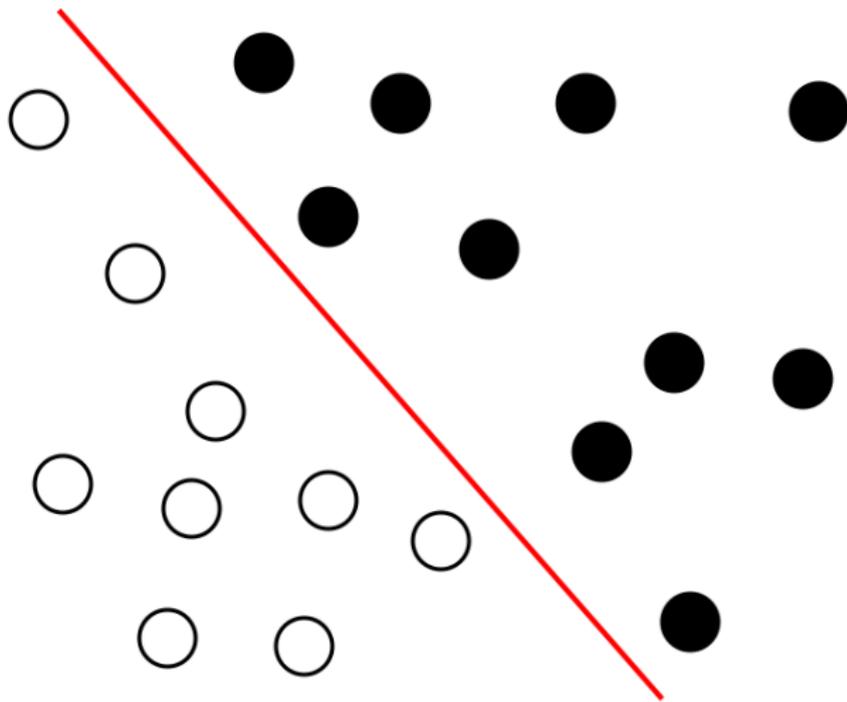
These slides are due in large part to Jean-Philippe Vert, who gave the course on kernel methods from 2004 to 2015 at the master MVA at ENS Cachan.

- Along the years, the course has become more and more exhaustive and the slides are probably one of the best reference available on kernels.
- This is a course with a **fairly large amount of math**, but still accessible to computer scientists who have heard what is a Hilbert space (at least once in their life).

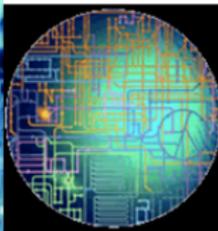
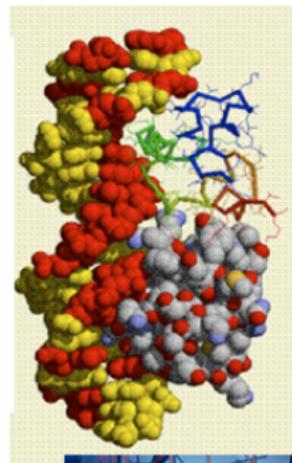
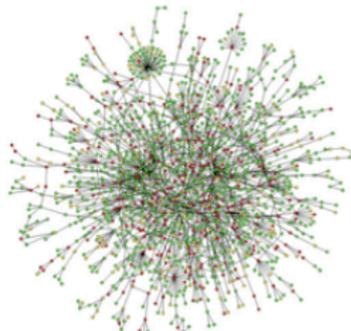
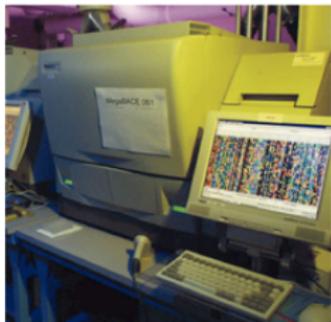
# Starting point: what we know is how to solve



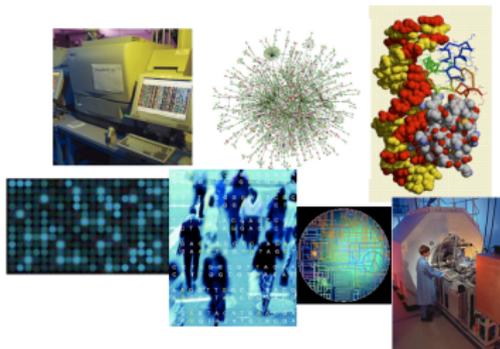
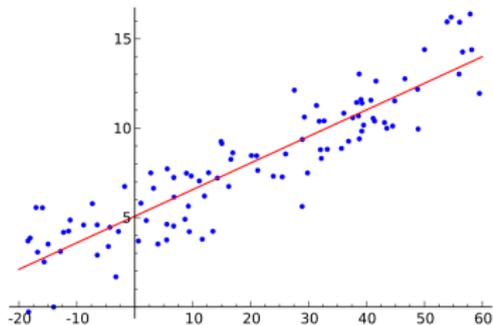
Or



But real data is often more complicated...



# Main goal of this course



- Extend well-understood, linear statistical learning techniques to real-world, complicated, structured, high-dimensional data (images, texts, time series, graphs, distributions, permutations...)

# A concrete supervised learning problem

## Regularized empirical risk formulation

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$



# A concrete supervised learning problem

## Regularized empirical risk formulation

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

A simple parametrization when  $\mathcal{X} = \mathbb{R}^p$  and  $\mathcal{Y} = \{-1, +1\}$ .

- $\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^p\}$  where the  $f_{\mathbf{w}}$ 's are linear:  $f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{x}^T \mathbf{w}$ .
- The regularization is the simple Euclidean norm  $\Omega(f_{\mathbf{w}}) = \|\mathbf{w}\|_2^2$ .

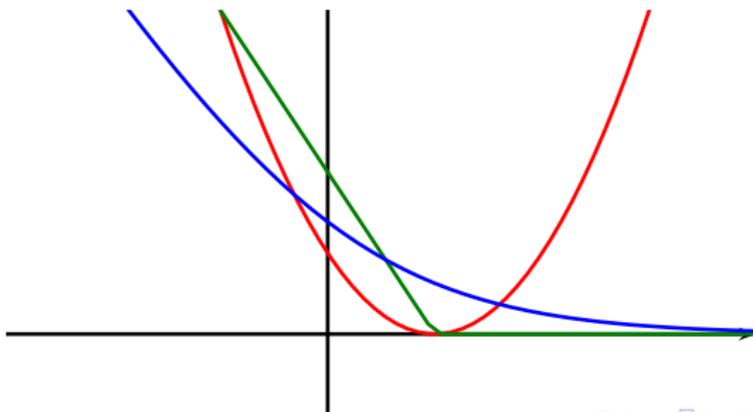
# A concrete supervised learning problem

This simple setting corresponds to many well-studied formulations.

**Ridge regression:** 
$$\min_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (\mathbf{y}_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2.$$

**Linear SVM:** 
$$\min_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \max(0, 1 - \mathbf{y}_i \mathbf{w}^\top \mathbf{x}_i) + \lambda \|\mathbf{w}\|_2^2.$$

**Logistic regression:** 
$$\min_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \log(1 + e^{-\mathbf{y}_i \mathbf{w}^\top \mathbf{x}_i}) + \lambda \|\mathbf{w}\|_2^2.$$



## A concrete supervised learning problem

Unfortunately, linear models often perform poorly unless the problem features are well-engineered or the problem is very simple.

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

First approach to work with a non-linear functional space  $\mathcal{F}$

- The “deep learning” space  $\mathcal{F}$  is parametrized:

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k \sigma_{k-1}(\mathbf{A}_{k-1} \dots \sigma_2(\mathbf{A}_2 \sigma_1(\mathbf{A}_1 \mathbf{x})) \dots)).$$

- Finding the optimal  $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k$  yields an (intractable) **non-convex** optimization problem in **huge dimension**.

# A concrete supervised learning problem

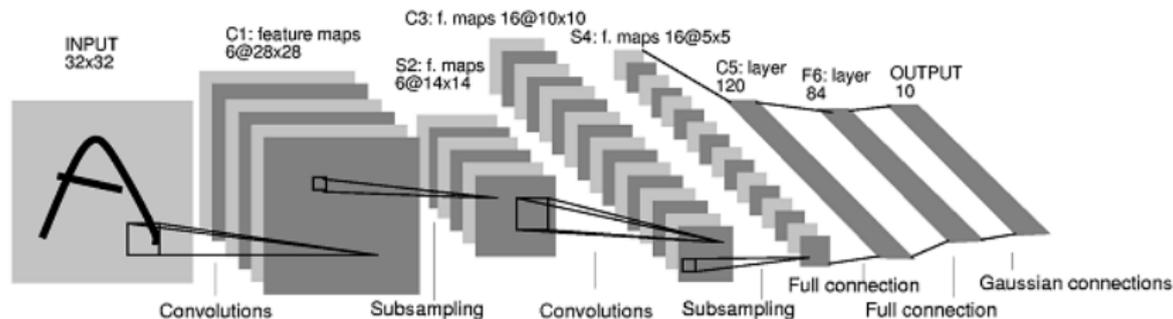


Figure: Exemple of convolutional neural network from LeCun et al. [1998]

## What are the main limitations of neural networks?

- Poor theoretical understanding.
- They require cumbersome hyper-parameter tuning.
- They are hard to regularize.

Despite these shortcomings, they have had an enormous success, thanks to large amounts of labeled data, computational power and engineering.

# A concrete supervised learning problem

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

## Second approach based on kernels

- Works with possibly infinite-dimensional functional spaces  $\mathcal{F}$ ;
- Works with non-vectorial structured data sets  $\mathcal{X}$  such as graphs;
- Regularization is natural and easy.

## Current limitations (and open research topics)

- Lack of scalability with  $n$  (traditionally  $O(n^2)$ );
- Lack of adaptivity to data and task.

# Organization of the course

## Content

- 1 Present the **basic theory** of kernel methods.
- 2 Develop a working knowledge of **kernel engineering** for specific data and applications (graphs, biological sequences, images).

## Practical

- Course homepage with slides, schedules, homework's etc...:  
<http://lear.inrialpes.fr/people/mairal/teaching/2015-2016/MSIAM/>.
- Evaluation: 50% homework + 50% data challenge.

# Outline

- 1 Kernels and RKHS
  - Kernels
  - Reproducing Kernel Hilbert Spaces (RKHS)
  - My first kernels
  - Smoothness functional
  - The kernel trick

# Outline

## 1 Kernels and RKHS

- Kernels
- Reproducing Kernel Hilbert Spaces (RKHS)
- My first kernels
- Smoothness functional
- The kernel trick

## 2 Kernels Methods: Supervised Learning

- The representer theorem
- Kernel ridge regression
- Classification with empirical risk minimization
- A (tiny) bit of learning theory
- Support vector machines

# Outline

- 3 Kernels Methods: Unsupervised Learning
  - Kernel K-means and spectral clustering
  - Kernel PCA
  - A quick note on kernel CCA

# Outline

- 3 Kernels Methods: Unsupervised Learning
  - Kernel K-means and spectral clustering
  - Kernel PCA
  - A quick note on kernel CCA
- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - Kernels on graphs

# Kernels and RKHS

# Overview

## Motivations

- Develop **versatile** algorithms to process and analyze data...
- ...without making any hypothesis regarding the **type of data** (vectors, strings, **graphs**, images, ...)

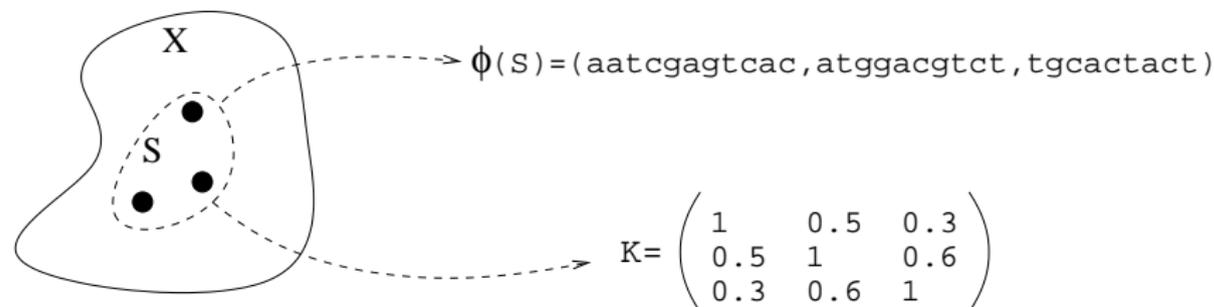
## The approach

- Develop methods based on **pairwise comparisons**.
- By imposing constraints on the pairwise comparison function (positive definite kernels), we obtain a **general framework for learning from data** (optimization in RKHS).

# Outline

- 1 Kernels and RKHS
  - Kernels
    - Reproducing Kernel Hilbert Spaces (RKHS)
    - My first kernels
    - Smoothness functional
    - The kernel trick
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

## Representation by pairwise comparisons



### Idea

- Define a “comparison function”:  $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ .
- Represent a set of  $n$  data points  $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  by the  $n \times n$  matrix:

$$[\mathbf{K}]_{ij} := K(\mathbf{x}_i, \mathbf{x}_j)$$

# Representation by pairwise comparisons

## Remarks

- Always a  $n \times n$  matrix, whatever the nature of data: **the same algorithm will work for any type of data** (vectors, strings, ...).
- Total **modularity** between the **choice of  $K$**  and the **choice of the algorithm**.
- **Poor scalability** w.r.t to the dataset size ( $n^2$ )
- We will restrict ourselves to a **particular class** of pairwise comparison functions.

# Positive Definite (p.d.) Kernels

## Definition

A **positive definite (p.d.) kernel** on the set  $\mathcal{X}$  is a function  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  **symmetric**:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}),$$

and which satisfies, for all  $N \in \mathbb{N}$ ,  $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \in \mathcal{X}^N$  and  $(a_1, a_2, \dots, a_N) \in \mathbb{R}^N$ :

$$\sum_{i=1}^N \sum_{j=1}^N a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

# Similarity matrices of p.d. kernels

## Remarks

- Equivalently, a kernel  $K$  is p.d. if and only if, for any  $N \in \mathbb{N}$  and any set of points  $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \in \mathcal{X}^N$ , the **similarity matrix**  $[\mathbf{K}]_{ij} := K(\mathbf{x}_i, \mathbf{x}_j)$  is **positive semidefinite**.
- **Kernel methods** are algorithms that take such matrices as input.

# The simplest p.d. kernel

## Lemma

Let  $\mathcal{X} = \mathbb{R}^d$ . The function  $K : \mathcal{X}^2 \mapsto \mathbb{R}$  defined by:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^d}$$

is p.d. (it is often called the **linear kernel**).

# The simplest p.d. kernel

## Lemma

Let  $\mathcal{X} = \mathbb{R}^d$ . The function  $K : \mathcal{X}^2 \mapsto \mathbb{R}$  defined by:

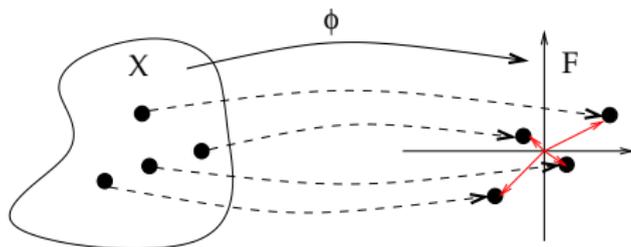
$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^d}$$

is p.d. (it is often called the **linear kernel**).

## Proof

- $\langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^d} = \langle \mathbf{x}', \mathbf{x} \rangle_{\mathbb{R}^d}$ ,
- $\sum_{i=1}^N \sum_{j=1}^N a_i a_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathbb{R}^d} = \left\| \sum_{i=1}^N a_i \mathbf{x}_i \right\|_{\mathbb{R}^d}^2 \geq 0$

## A more ambitious p.d. kernel

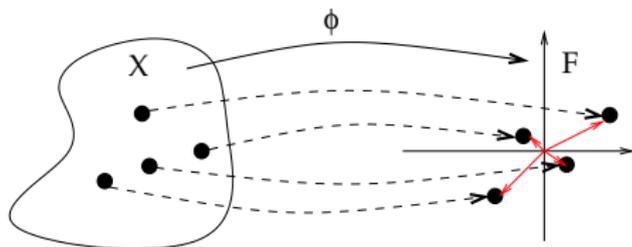


### Lemma

Let  $\mathcal{X}$  be any set, and  $\Phi : \mathcal{X} \mapsto \mathbb{R}^d$ . Then, the function  $K : \mathcal{X}^2 \mapsto \mathbb{R}$  defined as follows is p.d.:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathbb{R}^d} .$$

## A more ambitious p.d. kernel



### Lemma

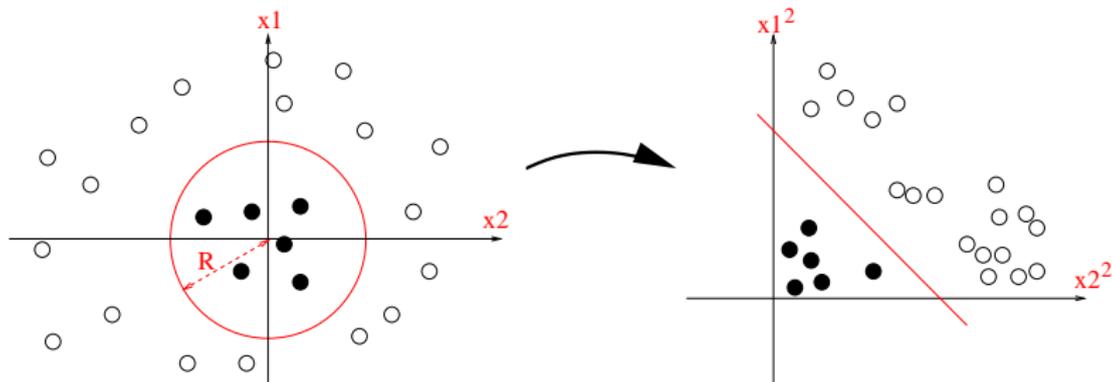
Let  $\mathcal{X}$  be any set, and  $\Phi : \mathcal{X} \mapsto \mathbb{R}^d$ . Then, the function  $K : \mathcal{X}^2 \mapsto \mathbb{R}$  defined as follows is p.d.:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathbb{R}^d} .$$

### Proof

- $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathbb{R}^d} = \langle \Phi(\mathbf{x}'), \Phi(\mathbf{x}) \rangle_{\mathbb{R}^d}$  ,
- $\sum_{i=1}^N \sum_{j=1}^N a_i a_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathbb{R}^d} = \left\| \sum_{i=1}^N a_i \Phi(\mathbf{x}_i) \right\|_{\mathbb{R}^d}^2 \geq 0$  .

## Example: polynomial kernel



For  $\vec{x} = (x_1, x_2)^T \in \mathbb{R}^2$ , let  $\vec{\Phi}(\vec{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$ :

$$\begin{aligned}K(\vec{x}, \vec{x}') &= x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2 x_2'^2 \\ &= (x_1x_1' + x_2x_2')^2 \\ &= (\vec{x} \cdot \vec{x}')^2.\end{aligned}$$

*Exercise: show that  $(\vec{x} \cdot \vec{x}')^d$  is p.d. for any integer  $d$ .*

# Conversely: Kernels as Inner Products

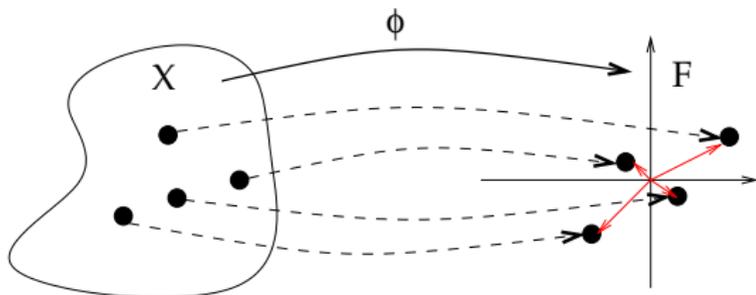
## Theorem (Aronszajn, 1950)

$K$  is a p.d. kernel on the set  $\mathcal{X}$  *if and only if* there exists a *Hilbert space*  $\mathcal{H}$  and a mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H},$$

such that, for any  $\mathbf{x}, \mathbf{x}'$  in  $\mathcal{X}$ :

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}.$$



# In case of ...

## Definitions

- An **inner product** on an  $\mathbb{R}$ -vector space  $\mathcal{H}$  is a mapping  $(f, g) \mapsto \langle f, g \rangle_{\mathcal{H}}$  from  $\mathcal{H}^2$  to  $\mathbb{R}$  that is **bilinear**, **symmetric** and such that  $\langle f, f \rangle > 0$  for all  $f \in \mathcal{H} \setminus \{0\}$ .
- A vector space endowed with an inner product is called **pre-Hilbert**. It is endowed with a norm defined as  $\|f\|_{\mathcal{H}} = \langle f, f \rangle_{\mathcal{H}}^{\frac{1}{2}}$ .
- A **Hilbert space** is a pre-Hilbert space **complete** for the norm  $\|\cdot\|_{\mathcal{H}}$ . That is, any Cauchy sequence in  $\mathcal{H}$  converges in  $\mathcal{H}$ .
- A **Cauchy sequence**  $(f_n)_{n \geq 0}$  is a sequence whose elements become progressively arbitrarily close to each other:

$$\lim_{N \rightarrow +\infty} \sup_{n, m \geq N} \|f_n - f_m\|_{\mathcal{H}} = 0.$$

Completeness is necessary to keep “good” convergence properties of Euclidean spaces in an infinite-dimensional context.

## Proof: finite case

### Proof

- Assume  $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  is finite of size  $N$ .
- Any p.d. kernel  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is entirely defined by the  $N \times N$  symmetric positive semidefinite matrix  $[\mathbf{K}]_{ij} := K(\mathbf{x}_i, \mathbf{x}_j)$ .
- It can therefore be diagonalized on an orthonormal basis of eigenvectors  $(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N)$ , with non-negative eigenvalues  $0 \leq \lambda_1 \leq \dots \leq \lambda_N$ , i.e.,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left[ \sum_{l=1}^N \lambda_l \mathbf{u}_l \mathbf{u}_l^T \right]_{ij} = \sum_{l=1}^N \lambda_l \mathbf{u}_l(i) \mathbf{u}_l(j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathbb{R}^N},$$

with

$$\Phi(\mathbf{x}_i) = \begin{pmatrix} \sqrt{\lambda_1} u_1(i) \\ \vdots \\ \sqrt{\lambda_N} u_N(i) \end{pmatrix}. \quad \square$$

## Proof: general case

- Mercer (1909) for  $\mathcal{X} = [a, b] \subset \mathbb{R}$  (more generally  $\mathcal{X}$  compact) and  $K$  continuous.
- Kolmogorov (1941) for  $\mathcal{X}$  countable.
- Aronszajn (1944, 1950) for the general case.

We will go through the proof of the general case by introducing the concept of Reproducing Kernel Hilbert Spaces (RKHS).

# Outline

- 1 Kernels and RKHS
  - Kernels
  - Reproducing Kernel Hilbert Spaces (RKHS)
  - My first kernels
  - Smoothness functional
  - The kernel trick
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

# RKHS Definition

## Definition

Let  $\mathcal{X}$  be a set and  $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$  be a **class of functions forming a (real) Hilbert space** with inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ . The function  $K : \mathcal{X}^2 \mapsto \mathbb{R}$  is called a **reproducing kernel (r.k.)** of  $\mathcal{H}$  if

- 1  $\mathcal{H}$  contains all functions of the form

$$\forall \mathbf{x} \in \mathcal{X}, \quad K_{\mathbf{x}} : \mathbf{t} \mapsto K(\mathbf{x}, \mathbf{t}) .$$

- 2 For every  $\mathbf{x} \in \mathcal{X}$  and  $f \in \mathcal{H}$  the **reproducing property** holds:

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}} .$$

If a r.k. exists, then  $\mathcal{H}$  is called a **reproducing kernel Hilbert space (RKHS)**.

# An equivalent definition of RKHS

## Theorem

The Hilbert space  $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$  is a RKHS if and only if for any  $\mathbf{x} \in \mathcal{X}$ , the mapping:

$$\begin{aligned} F : \mathcal{H} &\rightarrow \mathbb{R} \\ f &\mapsto f(\mathbf{x}) \end{aligned}$$

is **continuous**.

# An equivalent definition of RKHS

## Theorem

The Hilbert space  $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$  is a RKHS if and only if for any  $\mathbf{x} \in \mathcal{X}$ , the mapping:

$$\begin{aligned} F: \mathcal{H} &\rightarrow \mathbb{R} \\ f &\mapsto f(\mathbf{x}) \end{aligned}$$

is **continuous**.

## Corollary

**Convergence in a RKHS implies pointwise convergence**, i.e., if  $(f_n)_{n \in \mathbb{N}}$  converges to  $f$  in  $\mathcal{H}$ , then  $(f_n(\mathbf{x}))_{n \in \mathbb{N}}$  converges to  $f(\mathbf{x})$  for any  $\mathbf{x} \in \mathcal{X}$ .

# Proof

If  $\mathcal{H}$  is a RKHS then  $f \mapsto f(\mathbf{x})$  is continuous

If a r.k.  $K$  exists, then for any  $(\mathbf{x}, f) \in \mathcal{X} \times \mathcal{H}$ :

$$\begin{aligned} |f(\mathbf{x})| &= |\langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}}| \\ &\leq \|f\|_{\mathcal{H}} \cdot \|K_{\mathbf{x}}\|_{\mathcal{H}} \quad (\text{Cauchy-Schwarz}) \\ &\leq \|f\|_{\mathcal{H}} \cdot K(\mathbf{x}, \mathbf{x})^{\frac{1}{2}}, \end{aligned}$$

because  $\|K_{\mathbf{x}}\|_{\mathcal{H}}^2 = \langle K_{\mathbf{x}}, K_{\mathbf{x}} \rangle_{\mathcal{H}} = K(\mathbf{x}, \mathbf{x})$ . Therefore  $f \in \mathcal{H} \mapsto f(\mathbf{x}) \in \mathbb{R}$  is a continuous linear mapping.  $\square$

Since  $f$  is linear, it is indeed sufficient to show that  $f \rightarrow 0 \Rightarrow f(\mathbf{x}) \rightarrow 0$ .

## Proof (Converse)

If  $f \mapsto f(\mathbf{x})$  is continuous then  $\mathcal{H}$  is a RKHS

Conversely, let us assume that for any  $\mathbf{x} \in \mathcal{X}$  the linear form  $f \in \mathcal{H} \mapsto f(\mathbf{x})$  is continuous.

Then by Riesz representation theorem (general property of Hilbert spaces) there exists a unique  $g_{\mathbf{x}} \in \mathcal{H}$  such that:

$$f(\mathbf{x}) = \langle f, g_{\mathbf{x}} \rangle_{\mathcal{H}}$$

The function  $K(\mathbf{x}, \mathbf{y}) = g_{\mathbf{x}}(\mathbf{y})$  is then a r.k. for  $\mathcal{H}$ .  $\square$

# Unicity of r.k. and RKHS

## Theorem

- If  $\mathcal{H}$  is a RKHS, then it has a unique r.k.
- Conversely, a function  $K$  can be the r.k. of at most one RKHS.

# Unicity of r.k. and RKHS

## Theorem

- If  $\mathcal{H}$  is a RKHS, then it has a unique r.k.
- Conversely, a function  $K$  can be the r.k. of at most one RKHS.

## Consequence

This shows that we can talk of "the" kernel of a RKHS, or "the" RKHS of a kernel.

# Proof

If a r.k. exists then it is unique

Let  $K$  and  $K'$  be two r.k. of a RKHS  $\mathcal{H}$ . Then for any  $\mathbf{x} \in \mathcal{X}$ :

$$\begin{aligned}\|K_{\mathbf{x}} - K'_{\mathbf{x}}\|_{\mathcal{H}}^2 &= \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K_{\mathbf{x}} - K'_{\mathbf{x}} \rangle_{\mathcal{H}} \\ &= \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K_{\mathbf{x}} \rangle_{\mathcal{H}} - \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K'_{\mathbf{x}} \rangle_{\mathcal{H}} \\ &= K_{\mathbf{x}}(\mathbf{x}) - K'_{\mathbf{x}}(\mathbf{x}) - K_{\mathbf{x}}(\mathbf{x}) + K'_{\mathbf{x}}(\mathbf{x}) \\ &= 0.\end{aligned}$$

This shows that  $K_{\mathbf{x}} = K'_{\mathbf{x}}$  as functions, i.e.,  $K_{\mathbf{x}}(\mathbf{y}) = K'_{\mathbf{x}}(\mathbf{y})$  for any  $\mathbf{y} \in \mathcal{X}$ . In other words,  $K=K'$ .  $\square$

# Proof

If a r.k. exists then it is unique

Let  $K$  and  $K'$  be two r.k. of a RKHS  $\mathcal{H}$ . Then for any  $\mathbf{x} \in \mathcal{X}$ :

$$\begin{aligned}\|K_{\mathbf{x}} - K'_{\mathbf{x}}\|_{\mathcal{H}}^2 &= \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K_{\mathbf{x}} - K'_{\mathbf{x}} \rangle_{\mathcal{H}} \\ &= \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K_{\mathbf{x}} \rangle_{\mathcal{H}} - \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K'_{\mathbf{x}} \rangle_{\mathcal{H}} \\ &= K_{\mathbf{x}}(\mathbf{x}) - K'_{\mathbf{x}}(\mathbf{x}) - K_{\mathbf{x}}(\mathbf{x}) + K'_{\mathbf{x}}(\mathbf{x}) \\ &= 0.\end{aligned}$$

This shows that  $K_{\mathbf{x}} = K'_{\mathbf{x}}$  as functions, i.e.,  $K_{\mathbf{x}}(\mathbf{y}) = K'_{\mathbf{x}}(\mathbf{y})$  for any  $\mathbf{y} \in \mathcal{X}$ . In other words,  $K=K'$ .  $\square$

The RKHS of a r.k.  $K$  is unique

Left as exercise.

# An important result

## Theorem

A function  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is **p.d.** if and only if it is a **r.k.**

# Proof

## A r.k. is p.d.

- ① A r.k. is **symmetric** because, for any  $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2$ :

$$K(\mathbf{x}, \mathbf{y}) = \langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle_{\mathcal{H}} = \langle K_{\mathbf{y}}, K_{\mathbf{x}} \rangle_{\mathcal{H}} = K(\mathbf{y}, \mathbf{x}).$$

- ② It is **p.d.** because for any  $N \in \mathbb{N}, (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \in \mathcal{X}^N$ , and  $(a_1, a_2, \dots, a_N) \in \mathbb{R}^N$ :

$$\begin{aligned} \sum_{i,j=1}^N a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) &= \sum_{i,j=1}^N a_i a_j \langle K_{\mathbf{x}_i}, K_{\mathbf{x}_j} \rangle_{\mathcal{H}} \\ &= \left\| \sum_{i=1}^N a_i K_{\mathbf{x}_i} \right\|_{\mathcal{H}}^2 \\ &\geq 0. \quad \square \end{aligned}$$

# Proof

## A p.d. kernel is a r.k. (1/4)

- Let  $\mathcal{H}_0$  be the vector subspace of  $\mathbb{R}^{\mathcal{X}}$  spanned by the functions  $\{K_{\mathbf{x}}\}_{\mathbf{x} \in \mathcal{X}}$ .
- For any  $f, g \in \mathcal{H}_0$ , given by:

$$f = \sum_{i=1}^m a_i K_{\mathbf{x}_i}, \quad g = \sum_{j=1}^n b_j K_{\mathbf{y}_j},$$

let:

$$\langle f, g \rangle_{\mathcal{H}_0} := \sum_{i,j} a_i b_j K(\mathbf{x}_i, \mathbf{y}_j).$$

# Proof

A p.d. kernel is a r.k. (2/4)

- $\langle f, g \rangle_{\mathcal{H}_0}$  does not depend on the expansion of  $f$  and  $g$  because:

$$\langle f, g \rangle_{\mathcal{H}_0} = \sum_{i=1}^m a_i g(\mathbf{x}_i) = \sum_{j=1}^n b_j f(\mathbf{y}_j).$$

- This also shows that  $\langle \cdot, \cdot \rangle_{\mathcal{H}_0}$  is a **symmetric bilinear form**.
- This also shows that for any  $\mathbf{x} \in \mathcal{X}$  and  $f \in \mathcal{H}_0$ :

$$\langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}_0} = f(\mathbf{x}).$$

# Proof

## A p.d. kernel is a r.k. (3/4)

- $K$  is assumed to be p.d., therefore:

$$\|f\|_{\mathcal{H}_0}^2 = \sum_{i,j=1}^m a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

In particular Cauchy-Schwarz is valid with  $\langle \cdot, \cdot \rangle_{\mathcal{H}_0}$ .

- By Cauchy-Schwarz we deduce that  $\forall \mathbf{x} \in \mathcal{X}$ :

$$|f(\mathbf{x})| = |\langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}_0}| \leq \|f\|_{\mathcal{H}_0} \cdot K(\mathbf{x}, \mathbf{x})^{\frac{1}{2}},$$

therefore  $\|f\|_{\mathcal{H}_0} = 0 \implies f = 0$ .

- $\mathcal{H}_0$  is therefore a **pre-Hilbert space** endowed with the inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}_0}$ .

# Proof

## A p.d. kernel is a r.k. (4/4)

- For any Cauchy sequence  $(f_n)_{n \geq 0}$  in  $(\mathcal{H}_0, \langle \cdot, \cdot \rangle_{\mathcal{H}_0})$ , we note that:

$$\forall (\mathbf{x}, m, n) \in \mathcal{X} \times \mathbb{N}^2, \quad |f_m(\mathbf{x}) - f_n(\mathbf{x})| \leq \|f_m - f_n\|_{\mathcal{H}_0} \cdot K(\mathbf{x}, \mathbf{x})^{\frac{1}{2}}.$$

Therefore for any  $\mathbf{x}$  the sequence  $(f_n(\mathbf{x}))_{n \geq 0}$  is Cauchy in  $\mathbb{R}$  and has therefore a limit.

- If we add to  $\mathcal{H}_0$  the functions defined as the pointwise limits of Cauchy sequences, then the space becomes complete and is therefore a **Hilbert space**, with  $K$  as r.k. (up to a few technicalities, left as exercise).  $\square$

## Application: back to Aronszajn's theorem

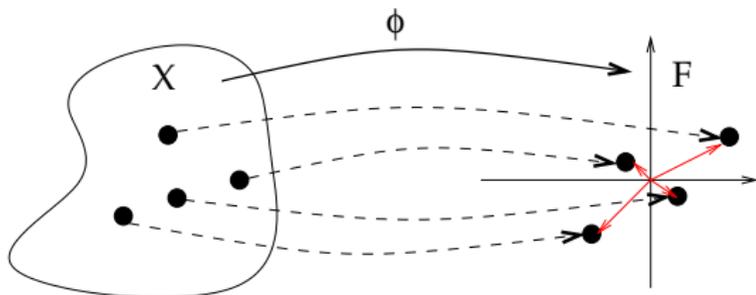
### Theorem (Aronszajn, 1950)

$K$  is a p.d. kernel on the set  $\mathcal{X}$  *if and only if* there exists a *Hilbert space*  $\mathcal{H}$  and a mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H},$$

such that, for any  $\mathbf{x}, \mathbf{x}'$  in  $\mathcal{X}$ :

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}.$$



# Proof of Aronzsajn's theorem

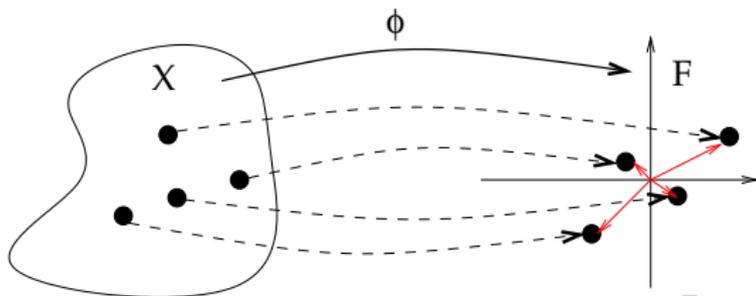
## Proof

- If  $K$  is p.d. over a set  $\mathcal{X}$  then it is the r.k. of a Hilbert space  $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ .
- Let the mapping  $\Phi : \mathcal{X} \rightarrow \mathcal{H}$  defined by:

$$\forall \mathbf{x} \in \mathcal{X}, \quad \Phi(\mathbf{x}) = K_{\mathbf{x}}.$$

- By the reproducing property we have:

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2, \quad \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}} = \langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle_{\mathcal{H}} = K(\mathbf{x}, \mathbf{y}). \quad \square$$



# Outline

- 1 Kernels and RKHS
  - Kernels
  - Reproducing Kernel Hilbert Spaces (RKHS)
  - **My first kernels**
  - Smoothness functional
  - The kernel trick
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

# The linear kernel

Take  $\mathcal{X} = \mathbb{R}^d$  and the **linear kernel**:

$$K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^d}$$

## Theorem

*The RKHS of the linear kernel is the set of linear functions of the form*

$$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle_{\mathbb{R}^d} \quad \text{for } \mathbf{w} \in \mathbb{R}^d,$$

*endowed with the norm*

$$\|f_{\mathbf{w}}\|_{\mathcal{H}} = \|\mathbf{w}\|_2.$$

# Proof

- The RKHS of the linear kernel consists of functions:

$$\mathbf{x} \in \mathbb{R}^d \mapsto f(\mathbf{x}) = \sum_i a_i \langle \mathbf{x}_i, \mathbf{x} \rangle_{\mathbb{R}^d} = \langle \mathbf{w}, \mathbf{x} \rangle_{\mathbb{R}^d},$$

with  $\mathbf{w} = \sum_i a_i \mathbf{x}_i$ .

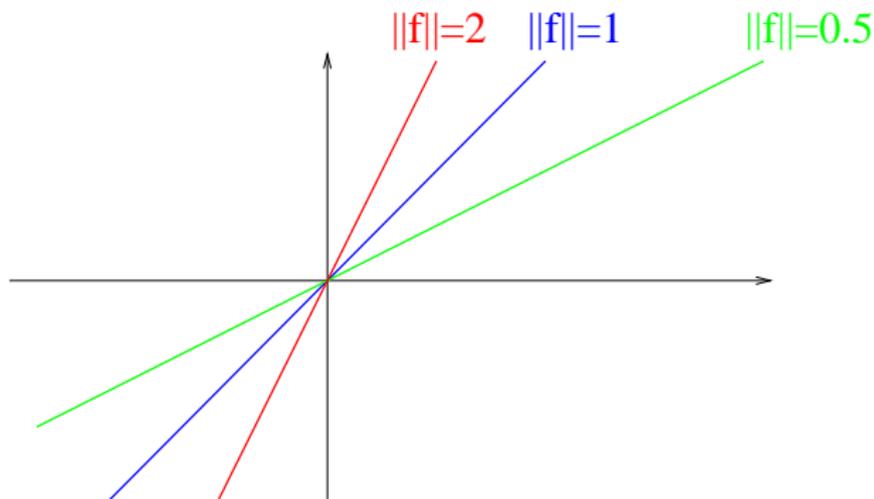
- The RKHS is therefore the set of **linear forms** endowed with the following inner product:

$$\langle f, g \rangle_{\mathcal{H}_K} = \langle \mathbf{w}, \mathbf{v} \rangle_{\mathbb{R}^d},$$

when  $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$  and  $g(\mathbf{x}) = \mathbf{v} \cdot \mathbf{x}$ .

## RKHS of the linear kernel (cont.)

$$\begin{cases} K_{lin}(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^\top \mathbf{x}' . \\ f(\mathbf{x}) &= \mathbf{w}^\top \mathbf{x} , \\ \|f\|_{\mathcal{H}} &= \|\mathbf{w}\|_2 . \end{cases}$$



# The polynomial kernel

We have already mentioned a generalization of the linear kernel: the **polynomial kernel of degree  $p$** :

$$K_{poly}(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^d} + c)^p$$

Let us find its RKHS for  $p = 2$  and  $c = 0$ .

# The polynomial kernel

We have already mentioned a generalization of the linear kernel: the **polynomial kernel of degree  $p$** :

$$K_{poly}(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^d} + c)^p$$

Let us find its RKHS for  $p = 2$  and  $c = 0$ .

**First step: Look for an inner-product.**

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= \text{trace} \left( \mathbf{x}^\top \mathbf{y} \mathbf{x}^\top \mathbf{y} \right) \\ &= \text{trace} \left( \mathbf{y}^\top \mathbf{x} \mathbf{x}^\top \mathbf{y} \right) \\ &= \text{trace} \left( \mathbf{x} \mathbf{x}^\top \mathbf{y} \mathbf{y}^\top \right) \\ &= \left\langle \mathbf{x} \mathbf{x}^\top, \mathbf{y} \mathbf{y}^\top \right\rangle_F, \end{aligned}$$

where  $F$  is the Froebenius norm for matrices in  $\mathbb{R}^{d \times d}$ .

# The polynomial kernel

## Second step: propose a candidate RKHS.

We know that  $\mathcal{H}$  contains all the functions

$$f(\mathbf{x}) = \sum_i a_i K(\mathbf{x}_i, \mathbf{x}) = \sum_i a_i \langle \mathbf{x}_i \mathbf{x}_i^\top, \mathbf{x} \mathbf{x}^\top \rangle_{\mathbb{F}} = \left\langle \sum_i a_i \mathbf{x}_i \mathbf{x}_i^\top, \mathbf{x} \mathbf{x}^\top \right\rangle.$$

Any symmetric matrix in  $\mathbb{R}^{d \times d}$  may be decomposed as  $\sum_i a_i \mathbf{x}_i \mathbf{x}_i^\top$ . Our candidate RKHS  $\mathcal{H}$  will be the set of quadratic functions

$$f_{\mathbf{S}}(\mathbf{x}) = \langle \mathbf{S}, \mathbf{x} \mathbf{x}^\top \rangle_{\mathbb{F}} = \mathbf{x}^\top \mathbf{S} \mathbf{x} \quad \text{for } \mathbf{S} \in \mathcal{S}^{d \times d},$$

where  $\mathcal{S}^{d \times d}$  is the set of symmetric matrices in  $\mathbb{R}^{d \times d}$ , endowed with the inner-product  $\langle f_{\mathbf{S}_1}, f_{\mathbf{S}_2} \rangle_{\mathcal{H}} = \langle \mathbf{S}_1, \mathbf{S}_2 \rangle_{\mathbb{F}}$ .

# The polynomial kernel

## Third step: check that the candidate is a Hilbert space.

This step is trivial in the present case since it is easy to see that  $\mathcal{H}$  a Euclidean space. Sometimes, things are not so simple and we need to prove the completeness explicitly.

## Fourth step: check that $\mathcal{H}$ is the RKHS.

$\mathcal{H}$  contains all the functions  $K_{\mathbf{x}} : \mathbf{t} \mapsto K(\mathbf{x}, \mathbf{t}) = \langle \mathbf{xx}^{\top}, \mathbf{tt}^{\top} \rangle_{\mathbb{F}}$ .

Moreover, we have for all  $f_{\mathbf{S}}$  in  $\mathcal{H}$  and  $\mathbf{x}$  in  $\mathcal{X}$ ,

$$f_{\mathbf{S}}(\mathbf{x}) = \langle \mathbf{S}, \mathbf{xx}^{\top} \rangle_{\mathbb{F}} = \langle f_{\mathbf{S}}, f_{\mathbf{xx}^{\top}} \rangle_{\mathcal{H}} = \langle f_{\mathbf{S}}, K_{\mathbf{x}} \rangle_{\mathcal{H}} \quad \square.$$

## Remark

All points  $\mathbf{x}$  in  $\mathcal{X}$  are mapped to a rank-one matrix  $\mathbf{xx}^{\top}$ . Most of points in  $\mathcal{H}$  do not admit a pre-image.

*Exercise: what is the RKHS of the general polynomial kernel?*

# Combining kernels

## Theorem

- If  $K_1$  and  $K_2$  are p.d. kernels, then:

$$K_1 + K_2,$$

$$K_1 K_2, \text{ and}$$

$$cK_1, \text{ for } c \geq 0,$$

are also p.d. kernels

- If  $(K_i)_{i \geq 1}$  is a sequence of p.d. kernels that converges pointwisely to a function  $K$ :

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \lim_{n \rightarrow \infty} K_n(\mathbf{x}, \mathbf{x}'),$$

then  $K$  is also a p.d. kernel.

*Proof: left as exercise*

# Examples

## Theorem

*If  $K$  is a kernel, then  $e^K$  is a kernel too.*

Proof:

$$e^{K(\mathbf{x}, \mathbf{x}')} = \lim_{n \rightarrow +\infty} \sum_{i=0}^n \frac{K(\mathbf{x}, \mathbf{x}')^i}{i!}$$

Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$

## Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$

## Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} + \mathbf{x}')$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} + \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} - \mathbf{x}')$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} + \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} - \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}')$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} + \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} - \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \max(\mathbf{x}, \mathbf{x}')$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1 - \mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x} + \mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} + \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} - \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \max(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}') / \max(\mathbf{x}, \mathbf{x}')$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} + \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} - \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \max(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}') / \max(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = \text{GCD}(\mathbf{x}, \mathbf{x}')$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} + \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} - \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \max(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}') / \max(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = \text{GCD}(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = \text{LCM}(\mathbf{x}, \mathbf{x}')$

## Quiz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$ ,  $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}+\mathbf{x}'}$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = 2^{\mathbf{x}\mathbf{x}'}$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \log(1 + \mathbf{x}\mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2)$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} + \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}$ ,  $K(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x} - \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \max(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{R}_+$ ,  $K(\mathbf{x}, \mathbf{x}') = \min(\mathbf{x}, \mathbf{x}') / \max(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = \text{GCD}(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = \text{LCM}(\mathbf{x}, \mathbf{x}')$
- $\mathcal{X} = \mathbb{N}$ ,  $K(\mathbf{x}, \mathbf{x}') = \text{GCD}(\mathbf{x}, \mathbf{x}') / \text{LCM}(\mathbf{x}, \mathbf{x}')$

# Outline

## 1 Kernels and RKHS

- Kernels
- Reproducing Kernel Hilbert Spaces (RKHS)
- My first kernels
- **Smoothness functional**
- The kernel trick

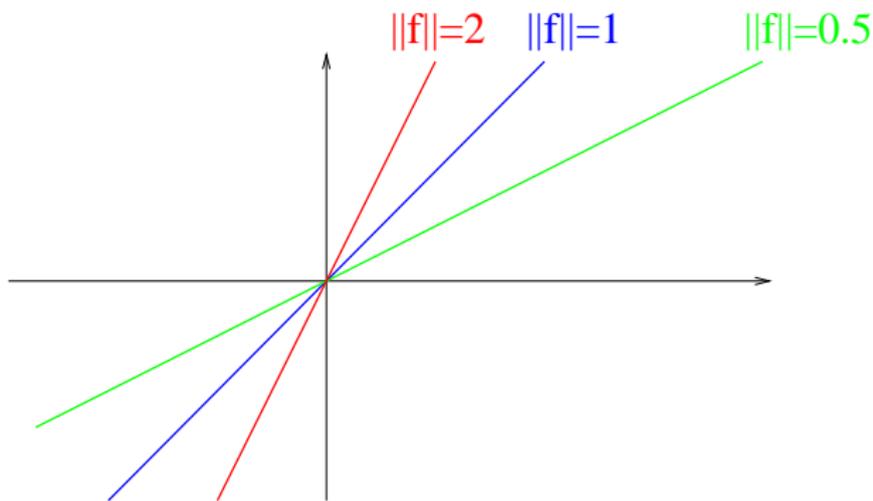
## 2 Kernels Methods: Supervised Learning

## 3 Kernels Methods: Unsupervised Learning

## 4 The Kernel Jungle

## Remember the RKHS of the linear kernel

$$\begin{cases} K_{lin}(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^\top \mathbf{x}' . \\ f(\mathbf{x}) &= \mathbf{w}^\top \mathbf{x} , \\ \|f\|_{\mathcal{H}} &= \|\mathbf{w}\|_2 . \end{cases}$$



# Smoothness functional

## A simple inequality

- By Cauchy-Schwarz we have, for any function  $f \in \mathcal{H}$  and any two points  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ :

$$\begin{aligned} |f(\mathbf{x}) - f(\mathbf{x}')| &= |\langle f, K_{\mathbf{x}} - K_{\mathbf{x}'} \rangle_{\mathcal{H}}| \\ &\leq \|f\|_{\mathcal{H}} \times \|K_{\mathbf{x}} - K_{\mathbf{x}'}\|_{\mathcal{H}} \\ &= \|f\|_{\mathcal{H}} \times d_K(\mathbf{x}, \mathbf{x}') . \end{aligned}$$

- The norm of a function in the RKHS controls **how fast** the function varies over  $\mathcal{X}$  with respect to the **geometry defined by the kernel** (Lipschitz with constant  $\|f\|_{\mathcal{H}}$ ).

## Important message

**Small norm  $\implies$  slow variations.**

## Kernels and RKHS : Summary

- P.d. kernels can be thought of as **inner product** after embedding the data space  $\mathcal{X}$  in some Hilbert space. As such a p.d. kernel defines a **metric** on  $\mathcal{X}$ .
- A realization of this embedding is the **RKHS**, valid without restriction on the space  $\mathcal{X}$  nor on the kernel.
- The RKHS is a space of functions over  $\mathcal{X}$ . The **norm** of a function in the RKHS is related to its degree of **smoothness** w.r.t. the metric defined by the kernel on  $\mathcal{X}$ .
- We will now see some applications of kernels and RKHS in statistics, before coming back to the problem of **choosing (and eventually designing) the kernel**.

# Outline

- 1 Kernels and RKHS
  - Kernels
  - Reproducing Kernel Hilbert Spaces (RKHS)
  - My first kernels
  - Smoothness functional
  - The kernel trick
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

# The kernel trick

- Choosing a p.d. kernel  $K$  on a set  $\mathcal{X}$  amounts to **embedding the data in a Hilbert space**: there exists a Hilbert space  $\mathcal{H}$  and a mapping  $\Phi : \mathcal{X} \mapsto \mathcal{H}$  such that, for all  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ ,

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}.$$

- However this mapping might **not be explicitly given**, nor convenient to work with in practice (e.g., large or even infinite dimensions).
- A solution is to work implicitly in the feature space!

## Kernel trick

**Any algorithm** to process finite-dimensional vectors that can be expressed **only in terms of pairwise inner products** can be applied to potentially infinite-dimensional vectors in the feature space of a p.d. kernel by **replacing each inner product evaluation by a kernel evaluation**.

# Kernel trick Summary

## Summary

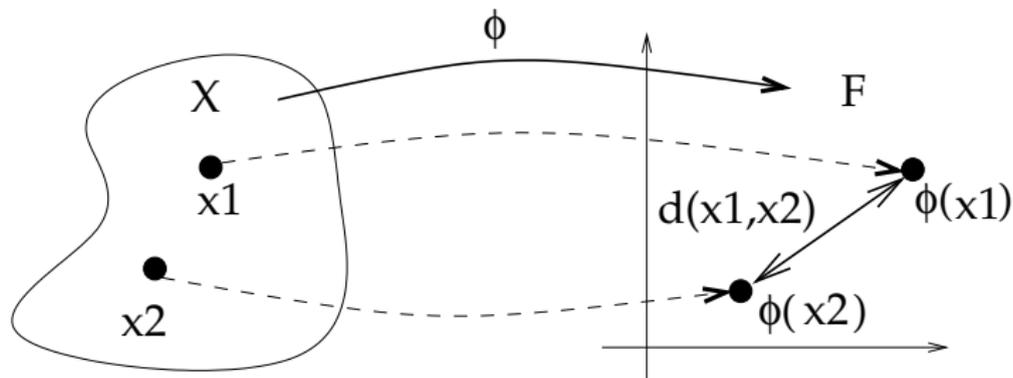
- The kernel trick is a trivial statement with **important applications**.
- It can be used to obtain **nonlinear** versions of well-known linear algorithms, e.g., by replacing the classical inner product by a Gaussian kernel.
- It can be used to apply classical algorithms to **non vectorial** data (e.g., strings, graphs) by again replacing the classical inner product by a valid kernel for the data.
- It allows in some cases to embed the initial space to a **larger feature space** and involve points in the feature space with no pre-image (e.g., barycenter).

## Examples from last week

Before moving to more complicated learning tasks, the kernel trick can be applied to simple elementary data manipulations in the feature space.

- 1 computing distances in the feature space;
- 2 computing distances between a point and a set;
- 3 computing the barycenter of points in the feature space;
- 4 centering points.

(see slides from previous lecture).



# Kernels Methods

## Supervised Learning

# Back to classifying cats and dogs

## Regularized empirical risk formulation

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$



# Back to classifying cats and dogs

## Regularized empirical risk formulation

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

A simple parametrization when  $\mathcal{X} = \mathbb{R}^p$  and  $\mathcal{Y} = \{-1, +1\}$ .

- $\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^p\}$  where the  $f_{\mathbf{w}}$ 's are linear:  $f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{x}^T \mathbf{w}$ .
- The regularization is the simple Euclidean norm  $\Omega(f_{\mathbf{w}}) = \|\mathbf{w}\|_2^2$ .

# Back to classifying cats and dogs

## Regularized empirical risk formulation

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

A simple parametrization when  $\mathcal{X} = \mathbb{R}^p$  and  $\mathcal{Y} = \{-1, +1\}$ .

This is equivalent to using a linear kernel  $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ .

In that case,  $\mathcal{F}$  is the Hilbert space  $\mathcal{H}$  of linear functions  $f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{x}^\top \mathbf{w}$  and  $\Omega(f_{\mathbf{w}}) = \|f_{\mathbf{w}}\|_{\mathcal{H}}^2 = \|\mathbf{w}\|_2^2$ .

# Back to classifying cats and dogs

## Regularized empirical risk formulation

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

## What are the new perspectives with kernel methods?

- being able to deal with non-linear functional spaces endowed with a natural regularization function  $\|\cdot\|_{\mathcal{H}}^2$ .
- being able to deal with non-vectorial data (graphs, trees).

# Motivations

Two theoretical results underpin a family of powerful algorithms for data analysis using positive definite kernels, collectively known as **kernel methods**:

- The **kernel trick**, based on the representation of p.d. kernels as inner products,
- the **representer theorem**, based on some properties of the regularization functional defined by the RKHS norm.

## An important property

When needed, the RKHS norm acts as a natural regularization function that penalizes variations of functions.

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
  - The representer theorem
  - Kernel ridge regression
  - Classification with empirical risk minimization
  - A (tiny) bit of learning theory
  - Support vector machines
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

# Back to classifying cats and dogs

## Regularized empirical risk formulation with kernels

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \|f\|_{\mathcal{H}}^2}_{\text{regularization}}. \quad (1)$$

**Question:** how to solve the above minimization problem?

- A simple theorem, called “representer theorem” can turn (1) into a concrete optimization problem in  $\mathbb{R}^n$ .

# The Theorem

## Representer Theorem

- Let  $\mathcal{X}$  be a set endowed with a p.d. kernel  $K$ ,  $\mathcal{H}_K$  the corresponding RKHS, and  $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathcal{X}$  a finite set of points in  $\mathcal{X}$ .
- Let  $\Psi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$  be a function of  $n + 1$  variables, strictly increasing with respect to the last variable.
- Then, any solution to the optimization problem:

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

admits a representation of the form:

$$\forall \mathbf{x} \in \mathcal{X}, \quad f(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}). \quad (3)$$

## Proof (1/2)

- Let  $\xi(f, \mathcal{S})$  be the functional that is minimized in the statement of the representer theorem, and  $\mathcal{H}_K^{\mathcal{S}}$  the linear span in  $\mathcal{H}_K$  of the vectors  $K_{x_i}$ , i.e.,

$$\mathcal{H}_K^{\mathcal{S}} = \left\{ f \in \mathcal{H}_K : f(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}), (\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n \right\}.$$

- $\mathcal{H}_K^{\mathcal{S}}$  finite-dimensional subspace, therefore any function  $f \in \mathcal{H}_K$  can be uniquely decomposed as:

$$f = f_{\mathcal{S}} + f_{\perp},$$

with  $f_{\mathcal{S}} \in \mathcal{H}_K^{\mathcal{S}}$  and  $f_{\perp} \perp \mathcal{H}_K^{\mathcal{S}}$  (by orthogonal projection).

## Proof (2/2)

- $\mathcal{H}_K$  being a RKHS it holds that:

$$\forall i = 1, \dots, n, \quad f_{\perp}(\mathbf{x}_i) = \langle f_{\perp}, K(\mathbf{x}_i, \cdot) \rangle_{\mathcal{H}_K} = 0,$$

because  $K(\mathbf{x}_i, \cdot) \in \mathcal{H}_K$ , therefore:

$$\forall i = 1, \dots, n, \quad f(\mathbf{x}_i) = f_S(\mathbf{x}_i).$$

- Pythagoras' theorem in  $\mathcal{H}_K$  then shows that:

$$\|f\|_{\mathcal{H}_K}^2 = \|f_S\|_{\mathcal{H}_K}^2 + \|f_{\perp}\|_{\mathcal{H}_K}^2.$$

- As a consequence,  $\xi(f, \mathcal{S}) \geq \xi(f_S, \mathcal{S})$ , with equality if and only if  $\|f_{\perp}\|_{\mathcal{H}_K} = 0$ . **The minimum of  $\Psi$  is therefore necessarily in  $\mathcal{H}_K^S$ .**

□

# Remarks

## Practical and theoretical consequences

Often the function  $\Psi$  has the form:

$$\Psi(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n), \|f\|_{\mathcal{H}_K}) = c(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)) + \lambda \Omega(\|f\|_{\mathcal{H}_K})$$

where  $c(\cdot)$  measures the “fit” of  $f$  to a given problem (regression, classification, dimension reduction, ...) and  $\Omega$  is strictly increasing. This formulation has two important consequences:

- **Theoretically**, the minimization will enforce the **norm**  $\|f\|_{\mathcal{H}_K}$  to be “small”, which can be beneficial by ensuring a sufficient level of smoothness for the solution (regularization effect).
- **Practically**, we know by the representer theorem that the solution lives in a **subspace of dimension  $n$** , which can lead to efficient algorithms although the RKHS itself can be of infinite dimension.

# Remarks

## Dual interpretations of kernel methods

Most kernel methods have two complementary interpretations:

- A **geometric interpretation** in the feature space, thanks to the kernel trick. Even when the feature space is “large”, most kernel methods work in the linear span of the embeddings of the points available.
- A **functional interpretation**, often as an optimization problem over (subsets of) the RKHS associated to the kernel.

The representer theorem has important consequences, but it is in fact rather trivial. We are looking for a function  $f$  in  $\mathcal{H}$  such that for all  $\mathbf{x}$  in  $\mathcal{X}$ ,  $f(\mathbf{x}) = \langle K_{\mathbf{x}}, f \rangle_{\mathcal{H}}$ . The part  $f^{\perp}$  that is orthogonal to the  $K_{x_i}$ 's is thus “useless” to explain the training data.

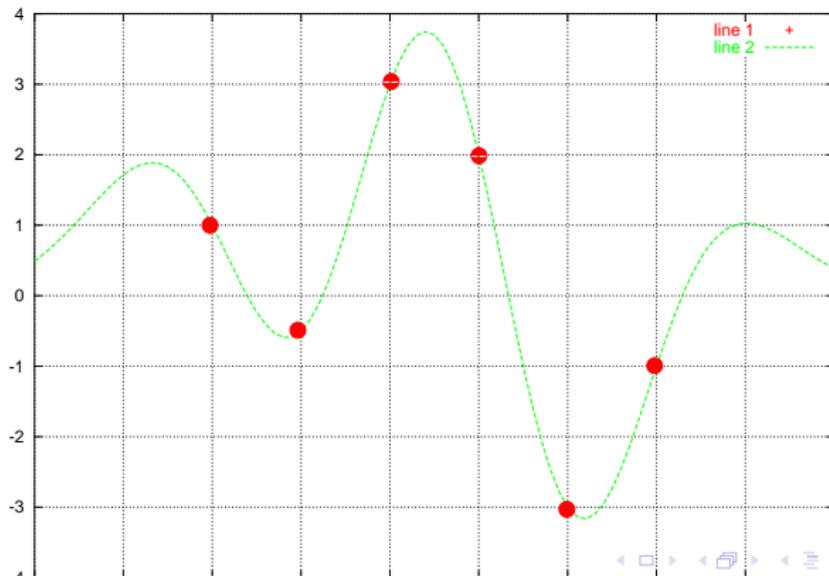
# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
  - The representer theorem
  - **Kernel ridge regression**
  - Classification with empirical risk minimization
  - A (tiny) bit of learning theory
  - Support vector machines
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

# Regression

## Setup

- Let  $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \in \mathcal{X}^n$  be a set of points
- Let  $\mathbf{y} = \{y_1, \dots, y_n\} \in \mathbb{R}^n$  be real numbers attached to the points
- Regression = find a function  $f : \mathcal{X} \rightarrow \mathbb{R}$  to **predict  $y$  by  $f(\mathbf{x})$**



# Least-square regression

- Let us quantify the error if  $f$  predicts  $f(\mathbf{x})$  instead of  $y$  by:

$$V(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2.$$

- Fix a set of functions  $\mathcal{H}$ .
- **Least-square regression** amounts to solving:

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

- Issues: unstable (especially in large dimensions), overfitting if  $\mathcal{H}$  is too “large”.

## Regularized least-square

- Let us consider a RKHS  $\mathcal{H}$ , RKHS associated to a p.d. kernel  $K$  on  $\mathcal{X}$ .
- Let us **regularize** the functional to be minimized by:

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

- *1st effect = **prevent overfitting** by penalizing non-smooth functions.*

## Representation of the solution

- By the representer theorem, any solution of:

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

can be expanded as:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}).$$

- *2nd effect = simplifying the solution.*

## Dual formulation

- Let  $\alpha = (\alpha_1, \dots, \alpha_n)^\top \in \mathbb{R}^n$ ,
- Let  $\mathbf{K}$  be the  $n \times n$  Gram matrix:  $\mathbf{K}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$ .
- We can then write:

$$\left( \hat{f}(\mathbf{x}_1), \dots, \hat{f}(\mathbf{x}_n) \right)^\top = \mathbf{K}\alpha,$$

- The following holds as usual:

$$\|\hat{f}\|_{\mathcal{H}_K}^2 = \alpha^\top \mathbf{K}\alpha.$$

# Dual formulation

- The problem is therefore equivalent to:

$$\arg \min_{\alpha \in \mathbb{R}^n} \frac{1}{n} (\mathbf{K}\alpha - \mathbf{y})^\top (\mathbf{K}\alpha - \mathbf{y}) + \lambda \alpha^\top \mathbf{K}\alpha.$$

- This is a convex and differentiable function of  $\alpha$ . Its minimum can therefore be found by setting the gradient in  $\alpha$  to zero:

$$\begin{aligned} 0 &= \frac{2}{n} \mathbf{K} (\mathbf{K}\alpha - \mathbf{y}) + 2\lambda \mathbf{K}\alpha \\ &= \mathbf{K} [(\mathbf{K} + \lambda n \mathbf{I}) \alpha - \mathbf{y}]. \end{aligned}$$

## Dual formulation

- $\mathbf{K}$  being a symmetric matrix, it can be diagonalized in an orthonormal basis and  $\text{Ker}(\mathbf{K}) \perp \text{Im}(\mathbf{K})$ .
- In this basis we see that  $(\mathbf{K} + \lambda nI)^{-1}$  leaves  $\text{Im}(\mathbf{K})$  and  $\text{Ker}(\mathbf{K})$  invariant.
- The problem is therefore equivalent to:

$$\begin{aligned} & (\mathbf{K} + \lambda nI) \boldsymbol{\alpha} - \mathbf{y} \in \text{Ker}(\mathbf{K}) \\ \Leftrightarrow & \boldsymbol{\alpha} - (\mathbf{K} + \lambda nI)^{-1} \mathbf{y} \in \text{Ker}(\mathbf{K}) \\ \Leftrightarrow & \boldsymbol{\alpha} = (\mathbf{K} + \lambda nI)^{-1} \mathbf{y} + \boldsymbol{\epsilon}, \text{ with } \mathbf{K}\boldsymbol{\epsilon} = 0. \end{aligned}$$

## Kernel ridge regression

- However, if  $\alpha' = \alpha + \epsilon$  with  $\mathbf{K}\epsilon = 0$ , then:

$$\|f - f'\|_{\mathcal{H}}^2 = (\alpha - \alpha')^\top \mathbf{K} (\alpha - \alpha') = 0,$$

therefore  $f = f'$ .

- One solution to the initial problem is therefore:

$$\hat{f} = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}),$$

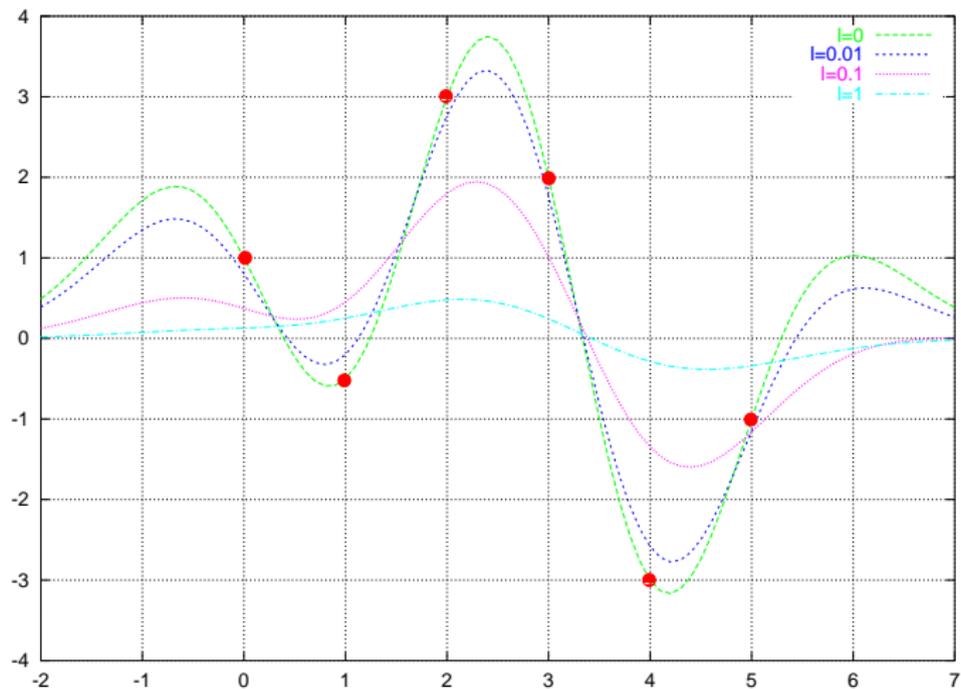
with

$$\alpha = (\mathbf{K} + \lambda nI)^{-1} \mathbf{y}.$$

## Remarks

- The matrix  $(\mathbf{K} + n\lambda I)^{-1}$  is **invertible when  $\lambda > 0$** .
- When  $\lambda \rightarrow 0$ , the method converges towards the solution of the classical unregularized least-square solution. When  $\lambda \rightarrow \infty$ , the solution converges to  $f = 0$ .
- In practice the symmetric matrix  $\mathbf{K} + n\lambda I$  is inverted with specific algorithms (e.g., Cholevsky decomposition).
- This method becomes difficult to use when the number of points becomes large.

# Example



## Kernel methods: Summary

- The **kernel trick** allows to extend many linear algorithms to **non-linear settings** and to **general data** (even non-vectorial).
- The **representer theorem** shows that that functional optimization over (subsets of) the RKHS is **feasible in practice**.
- We will see next a particularly successful applications of kernel methods, pattern recognition.

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
  - The representer theorem
  - Kernel ridge regression
  - **Classification with empirical risk minimization**
  - A (tiny) bit of learning theory
  - Support vector machines
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

# Pattern recognition



APPLE



APPLE



PEAR



PEAR



???



???



APPLE



PEAR



APPLE



APPLE



???

- **Input** variables  $\mathbf{x} \in \mathcal{X}$ .
- **Output**  $\mathbf{y} \in \{-1, 1\}$ .
- **Training set**  $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$ .

# Or again the cats and dogs example...

## Regularized empirical risk formulation

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$



...which we may reformulate with kernels

## Regularized empirical risk formulation

The goal is to learn a **prediction function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$  given labeled training data  $(\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})_{i=1, \dots, n}$ :

$$\min_{f \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{y}_i f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \|f\|_{\mathcal{H}}^2}_{\text{regularization}} .$$

By the **representer theorem**, the solution of the unconstrained problem can be expanded as:

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}) .$$

# Optimization in RKHS

- Plugging into the original problem we obtain the following **unconstrained and convex optimization problem in  $\mathbb{R}^n$** :

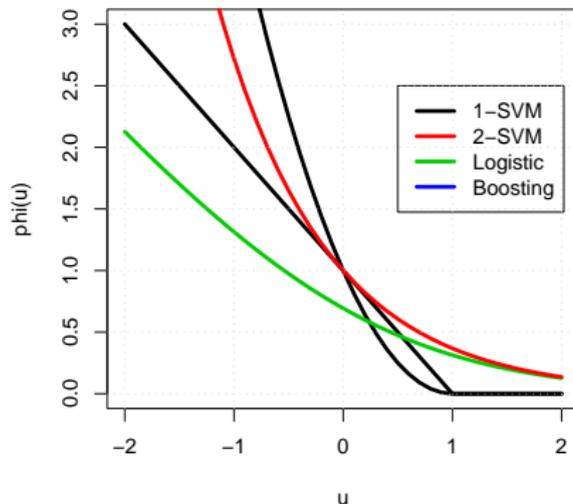
$$\min_{\alpha \in \mathbb{R}^n} \left\{ \frac{1}{n} \sum_{i=1}^n \varphi \left( \mathbf{y}_i \sum_{j=1}^n \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \right) + \lambda \sum_{i,j=1}^n \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \right\} .$$

- which in matrix notation gives

$$\min_{\alpha \in \mathbb{R}^n} \left\{ \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{y}_i [\mathbf{K}\alpha]_i) + \lambda \alpha^\top \mathbf{K} \alpha \right\} , .$$

- This can be **implemented** using general packages for **convex optimization** or specific algorithms (e.g., for SVM).

# Loss function examples



Method	$\varphi(u)$
Kernel logistic regression	$\log(1 + e^{-u})$
Support vector machine (1-SVM)	$\max(1 - u, 0)$
Support vector machine (2-SVM)	$\max(1 - u, 0)^2$
Boosting	$e^{-u}$

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
  - The representer theorem
  - Kernel ridge regression
  - Classification with empirical risk minimization
  - A (tiny) bit of learning theory
  - Support vector machines
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

# Formalization

## Definition of the risk and notation

- Let  $P$  be an (unknown) distribution on  $\mathcal{X} \times \mathcal{Y}$ .
- **Observation**:  $\mathcal{S}_n = (X_i, Y_i)_{i=1, \dots, n}$  i.i.d. random variables according to  $P$ .
- **Loss function**  $L(f(\mathbf{x}), \mathbf{y}) \in \mathbb{R}$  small when  $f(\mathbf{x})$  is a good predictor for  $y$ .
- **Risk**:  $R(f) = \mathbb{E}[L(f(X), Y)]$ .
- **Estimator**  $\hat{f}_n : \mathcal{X} \rightarrow \mathcal{Y}$ .
- **Goal**: small risk  $R(\hat{f}_n)$ .

# Large-margin classifiers

## Definition of the margin

- For pattern recognition  $\mathcal{Y} = \{-1, 1\}$ .
- The goal is to estimate a prediction function  $f : \mathcal{X} \rightarrow \mathbb{R}$ .
- The **margin** of the function  $f$  for a pair  $(\mathbf{x}, \mathbf{y})$  is:

$$\mathbf{y}f(\mathbf{x}).$$

## Large margin classifiers

- Focusing on **large margins** ensures that  $f(\mathbf{x})$  has the same sign as  $\mathbf{y}$  and a large absolute value (confidence).
- Suggests a loss function  $L(f(\mathbf{x}), \mathbf{y}) = \varphi(\mathbf{y}f(\mathbf{x}))$ , where  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  is **non-increasing**.
- **Goal**: small  $\varphi$ -risk  $R_\varphi(f) = \mathbb{E}[\varphi(Yf(X))]$ .

# Empirical risk minimization (ERM)

## ERM estimator

- Given  $n$  observations, the empirical  $\varphi$ -risk is:

$$R_{\varphi}^n(f) = \frac{1}{n} \sum_{i=1}^n \varphi(Y_i f(X_i)) .$$

- The **ERM estimator** on the functional class  $\mathcal{F}$  is the solution (when it exists) of:

$$\hat{f}_n = \arg \min_{f \in \mathcal{F}} R_{\varphi}^n(f) .$$

# Empirical risk minimization (ERM)

## ERM estimator

- Given  $n$  observations, the empirical  $\varphi$ -risk is:

$$R_{\varphi}^n(f) = \frac{1}{n} \sum_{i=1}^n \varphi(Y_i f(X_i)) .$$

- The **ERM estimator** on the functional class  $\mathcal{F}$  is the solution (when it exists) of:

$$\hat{f}_n = \arg \min_{f \in \mathcal{F}} R_{\varphi}^n(f) .$$

## Question

When is  $R_{\varphi}^n(f)$  a good estimate of the true risk  $R_{\varphi}(f)$ ?

# Class capacity

## Motivations

- The ERM principle gives a good solution if  $R_\varphi^n(\hat{f}_n)$  is similar to the minimum achievable risk  $\inf_{f \in \mathcal{F}} R_\varphi(f)$ .
- This can be ensured if  $\mathcal{F}$  is **not** “too large”.
- We need a measure of the “**capacity**” of  $\mathcal{F}$ .

## Definition: Rademacher complexity

The **Rademacher complexity** of a class of functions  $\mathcal{F}$  is:

$$\text{Rad}_n(\mathcal{F}) = \mathbb{E}_{X, \sigma} \left[ \sup_{f \in \mathcal{F}} \left| \frac{2}{n} \sum_{i=1}^n \sigma_i f(X_i) \right| \right],$$

where the expectation is over  $(X_i)_{i=1, \dots, n}$  and the independent uniform  $\{\pm 1\}$ -valued (Rademacher) random variables  $(\sigma_i)_{i=1, \dots, n}$ .

## Basic learning bounds

- Suppose  $\varphi$  is **Lipschitz** with constant  $L_\varphi$ :

$$\forall u, u' \in \mathbb{R}, \quad |\varphi(u) - \varphi(u')| \leq L_\varphi |u - u'|.$$

- Then on average over the training set (and with high probability) the  $\varphi$ -risk of the ERM estimator is closed to the empirical one:

$$\mathbb{E}_{\mathcal{S}} \left[ R_\varphi(\hat{f}_n) - R_\varphi^n(\hat{f}_n) \right] \leq 2L_\varphi \text{Rad}_n(\mathcal{F}).$$

- The  $\varphi$ -risk of the ERM estimator is also close to the smallest achievable on  $\mathcal{F}$  (on average and with large probability):

$$\mathbb{E}_{\mathcal{S}} R_\varphi(\hat{f}_n) \leq \inf_{f \in \mathcal{F}} R_\varphi(f) + 4L_\varphi \text{Rad}_n(\mathcal{F}).$$

# ERM in RKHS balls

## Principle

- Assume  $\mathcal{X}$  is endowed with a p.d. kernel.
- We consider the ball of radius  $B$  in the RKHS as function class for the ERM:

$$\mathcal{F}_B = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq B\}.$$

## Theorem (capacity control of RKHS balls)

$$\text{Rad}_n(\mathcal{F}_B) \leq \frac{2B\sqrt{\mathbb{E}K(X, X)}}{\sqrt{n}}.$$

# Proof (1/2)

$$\begin{aligned}\text{Rad}_n(\mathcal{F}_B) &= \mathbb{E}_{X,\sigma} \left[ \sup_{f \in \mathcal{F}_B} \left| \frac{2}{n} \sum_{i=1}^n \sigma_i f(X_i) \right| \right] \\ &= \mathbb{E}_{X,\sigma} \left[ \sup_{f \in \mathcal{F}_B} \left| \left\langle f, \frac{2}{n} \sum_{i=1}^n \sigma_i K_{X_i} \right\rangle \right| \right] \quad (\text{RKHS}) \\ &= \mathbb{E}_{X,\sigma} \left[ B \left\| \frac{2}{n} \sum_{i=1}^n \sigma_i K_{X_i} \right\|_{\mathcal{H}} \right] \quad (\text{Cauchy-Schwarz}) \\ &= \frac{2B}{n} \mathbb{E}_{X,\sigma} \left[ \sqrt{\left\| \sum_{i=1}^n \sigma_i K_{X_i} \right\|_{\mathcal{H}}^2} \right] \\ &\leq \frac{2B}{n} \sqrt{\mathbb{E}_{X,\sigma} \left[ \sum_{i,j=1}^n \sigma_i \sigma_j K(X_i, X_j) \right]} \quad (\text{Jensen})\end{aligned}$$

## Proof (2/2)

But  $\mathbb{E}_\sigma [\sigma_i \sigma_j]$  is 1 if  $i = j$ , 0 otherwise. Therefore:

$$\begin{aligned} \text{Rad}_n(\mathcal{F}_B) &\leq \frac{2B}{n} \sqrt{\mathbb{E}_X \left[ \sum_{i,j=1}^n \mathbb{E}_\sigma [\sigma_i \sigma_j] K(X_i, X_j) \right]} \\ &\leq \frac{2B}{n} \sqrt{\mathbb{E}_X \sum_{i=1}^n K(X_i, X_i)} \\ &= \frac{2B \sqrt{\mathbb{E}_X K(X, X)}}{\sqrt{n}}. \quad \square \end{aligned}$$

# Basic learning bounds in RKHS balls

## Corollary

- Suppose  $K(X, X) \leq \kappa^2$  a.s. (e.g., Gaussian kernel and  $\kappa = 1$ ).
- Let the minimum possible  $\varphi$ -risk:

$$R_\varphi^* = \inf_{f \text{ measurable}} R_\varphi(f).$$

- Then we directly get for the ERM estimator in  $\mathcal{F}_B$ :

$$\mathbb{E}R_\varphi(\hat{f}_n) - R_\varphi^* \leq \frac{8L_\varphi\kappa B}{\sqrt{n}} + \left[ \inf_{f \in \mathcal{F}_B} R_\varphi(f) - R_\varphi^* \right].$$

# Choice of $B$ by structural risk minimization

## Remark

- The **estimation error upper bound**  $8L_\varphi\kappa B/\sqrt{n}$  increases (linearly) with  $B$ .
- The **approximation error**  $[\inf_{f \in \mathcal{F}_B} R_\varphi(f) - R_\varphi^*]$  decreases with  $B$ .
- Ideally the choice of  $B$  should find a **trade-off** that minimizes the upper bound.
- This is achieved when

$$\frac{\partial \inf_{f \in \mathcal{F}_B} R_\varphi(f)}{\partial B} = -\frac{8L_\varphi\kappa}{\sqrt{n}}.$$

# ERM in practice

## Reformulation as penalized minimization

- We must solve the constrained minimization problem:

$$\begin{cases} \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{y}_i f(\mathbf{x}_i)) \\ \text{subject to } \|f\|_{\mathcal{H}} \leq B. \end{cases}$$

- This is a **constrained optimization problem**.

# ERM in practice

## Reformulation as penalized minimization

- We must solve the constrained minimization problem:

$$\begin{cases} \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{y}_i f(\mathbf{x}_i)) \\ \text{subject to } \|f\|_{\mathcal{H}} \leq B. \end{cases}$$

- This is a **constrained optimization problem**.
- To make this practical we assume that  $\varphi$  is **convex**.
- The problem is then a **convex problem** in  $f$  for which **strong duality holds**. In particular  $f$  solves the problem if and only if it solves for some dual parameter  $\lambda$  the **unconstrained problem**:

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

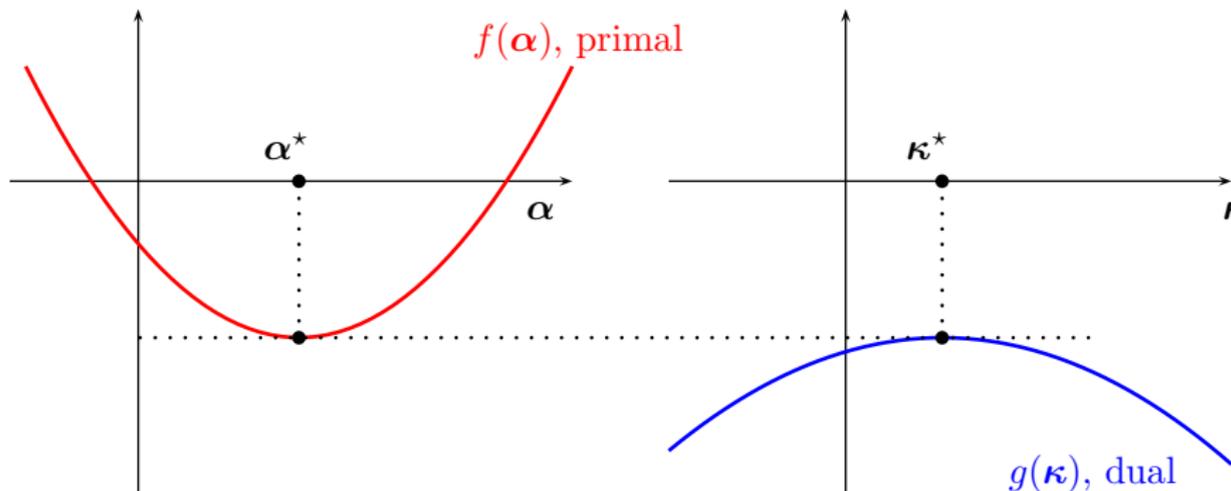
# One slide on convex duality

Following the course of Anatoli Juditsky would be useful here...

## One slide on convex duality

Following the course of Anatoli Juditsky would be useful here...

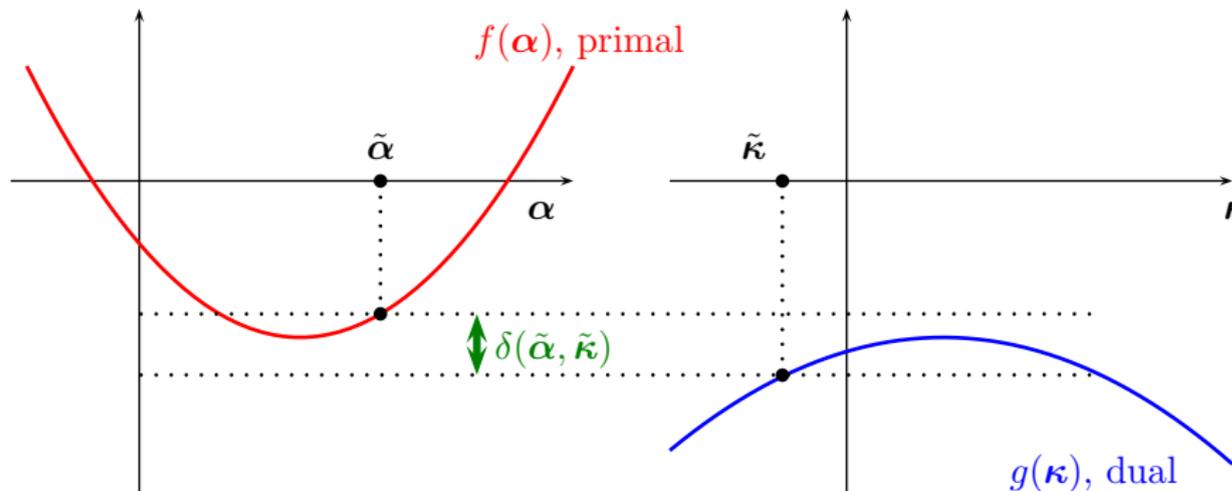
### Strong Duality



- The relation between  $\kappa^*$  and  $\alpha^*$  is not always known a priori.
- Very often, a constrained problem in the primal becomes a penalized one in the dual and vice-versae.

# Two slides on convex duality

## Parenthesis on duality gaps



- The duality gap guarantees us that  $0 \leq f(\tilde{\alpha}) - f(\alpha^*) \leq \delta(\tilde{\alpha}, \tilde{\kappa})$ .
- Dual problems are often obtained by Lagrangian or Fenchel duality.

# A few slides on Lagrangian duality

## Setting

- We consider an equality and inequality constrained optimization problem over a variable  $x \in \mathcal{X}$ :

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && h_i(x) = 0, \quad i = 1, \dots, m, \\ & && g_j(x) \leq 0, \quad j = 1, \dots, r, \end{aligned}$$

making **no assumption** of  $f$ ,  $g$  and  $h$ .

- Let us denote by  $f^*$  the optimal value of the decision function under the constraints, i.e.,  $f^* = f(x^*)$  if the minimum is reached at a global minimum  $x^*$ .

# A few slides on Lagrangian duality

## Lagrangian

The **Lagrangian** of this problem is the function  $L : \mathcal{X} \times \mathbb{R}^m \times \mathbb{R}^r \rightarrow \mathbb{R}$  defined by:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i h_i(x) + \sum_{j=1}^r \mu_j g_j(x).$$

## Lagrangian dual function

The **Lagrange dual function**  $g : \mathbb{R}^m \times \mathbb{R}^r \rightarrow \mathbb{R}$  is:

$$\begin{aligned} q(\lambda, \mu) &= \inf_{x \in \mathcal{X}} L(x, \lambda, \mu) \\ &= \inf_{x \in \mathcal{X}} \left( f(x) + \sum_{i=1}^m \lambda_i h_i(x) + \sum_{j=1}^r \mu_j g_j(x) \right). \end{aligned}$$

## A few slides on convex Lagrangian duality

For the (primal) problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h(x) = 0, \quad g(x) \leq 0, \end{array}$$

the **Lagrange dual problem** is:

$$\begin{array}{ll} \text{maximize} & q(\lambda, \mu) \\ \text{subject to} & \mu \geq 0, \end{array}$$

- $q$  is concave in  $(\lambda, \mu)$ , even if the original problem is not convex.
- when  $f$  is convex and strong duality holds, we have for all feasible primal and dual points  $x, \lambda, \mu$ ,

$$q(\lambda, \mu) \leq q(\lambda^*, \mu^*) = L(x^*, \lambda^*, \mu^*) = f(x^*) \leq f(x).$$

# Outline

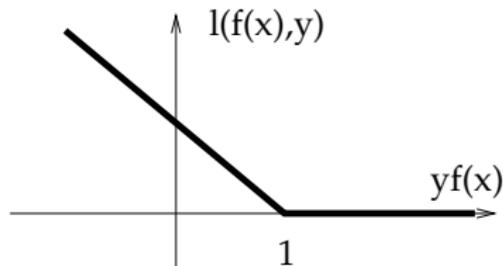
- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
  - The representer theorem
  - Kernel ridge regression
  - Classification with empirical risk minimization
  - A (tiny) bit of learning theory
  - Support vector machines
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle

# Motivations

## Support vector machines (SVM)

- Historically the first “kernel method” for pattern recognition, still the **most popular**.
- Often **state-of-the-art** in performance.
- One particular choice of loss function (**hinge loss**).
- Leads to a **sparse solution**, i.e., not all points are involved in the decomposition (**compression**).
- Particular algorithm for fast optimization (**decomposition by chunking methods**).

# Definitions



- The loss function is the **hinge loss**:

$$\varphi_{\text{hinge}}(u) = \max(1 - u, 0) = \begin{cases} 0 & \text{if } u \geq 1, \\ 1 - u & \text{otherwise.} \end{cases}$$

- SVM solve the problem:

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

# Problem reformulation (1/3)

## Slack variables

- This is a **convex** optimization problem
- However the objective function is not differentiable, so we reformulate the problem with additional **slack variables**  $\xi_1, \dots, \xi_n \in \mathbb{R}$ :

$$\min_{f \in \mathcal{H}, \boldsymbol{\xi} \in \mathbb{R}^n} \left\{ \frac{1}{n} \sum_{i=1}^n \xi_i + \lambda \|f\|_{\mathcal{H}}^2 \right\},$$

subject to:

$$\xi_i \geq \varphi_{\text{hinge}}(\mathbf{y}_i f(\mathbf{x}_i)).$$

## Problem reformulation (2/3)

The objective function is now differentiable in  $f$  and  $\xi_i$ , and we can rewrite the constraints as a conjunction of linear constraints:

$$\min_{f \in \mathcal{H}, \xi \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^n \xi_i + \lambda \|f\|_{\mathcal{H}}^2,$$

subject to:

$$\begin{cases} \xi_i \geq 1 - \mathbf{y}_i f(\mathbf{x}_i), & \text{for } i = 1, \dots, n, \\ \xi_i \geq 0, & \text{for } i = 1, \dots, n. \end{cases}$$

## Problem reformulation (3/3)

### Finite-dimensional expansion

Replacing  $\hat{f}$  by

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}),$$

the problem can be rewritten as an optimization problem in  $\alpha$  and  $\xi$ :

$$\min_{\alpha \in \mathbb{R}^n, \xi \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^n \xi_i + \lambda \alpha^\top \mathbf{K} \alpha,$$

subject to:

$$\begin{cases} \mathbf{y}_i \sum_{j=1}^n \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \xi_i - 1 \geq 0, & \text{for } i = 1, \dots, n, \\ \xi_i \geq 0, & \text{for } i = 1, \dots, n. \end{cases}$$

# Solving the problem

## Remarks

- This is a classical **quadratic program** (minimization of a convex quadratic function with linear constraints) for which any out-of-the-box optimization package can be used.
- The **dimension** of the problem and the **number of constraints**, however, are  $2n$  where  $n$  is the number of points. General-purpose QP solvers will have difficulties when  $n$  exceeds a few thousands.
- Solving the **dual** of this problem (also a QP) will be more convenient and lead to faster algorithms (due to the sparsity of the final solution).

# Dual problem

## With Lagrangian duality

- It can be shown that the dual problem is equivalent to:

$$\max_{0 \leq \mu \leq 1/n} \sum_{i=1}^n \mu_i - \frac{1}{4\lambda} \sum_{i,j=1}^n \mathbf{y}_i \mathbf{y}_j \mu_i \mu_j K(\mathbf{x}_i, \mathbf{x}_j).$$

*Proof: left as exercise*

## Back to the primal

- Once the dual problem is solved in  $\mu$  we get a solution of the primal problem by  $\alpha = Y\mu/2\lambda$ .
- We can therefore directly plug this into the dual problem to obtain the QP that  $\alpha$  must solve:

$$\max_{\alpha \in \mathbb{R}^d} 2 \sum_{i=1}^n \alpha_i y_i - \sum_{i,j=1}^n \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) = 2\alpha^\top \mathbf{y} - \alpha^\top \mathbf{K} \alpha,$$

subject to:

$$0 \leq y_i \alpha_i \leq \frac{1}{2\lambda n}, \quad \text{for } i = 1, \dots, n.$$

## Karush-Kuhn-Tucker (KKT) conditions

- The KKT optimality conditions are, for  $i = 1, \dots, n$ :

$$\begin{cases} \mu_i [y_i f(\mathbf{x}_i) + \xi_i - 1] = 0, \\ \nu_i \xi_i = 0, \end{cases}$$

- In terms of  $\alpha$  this can be rewritten as:

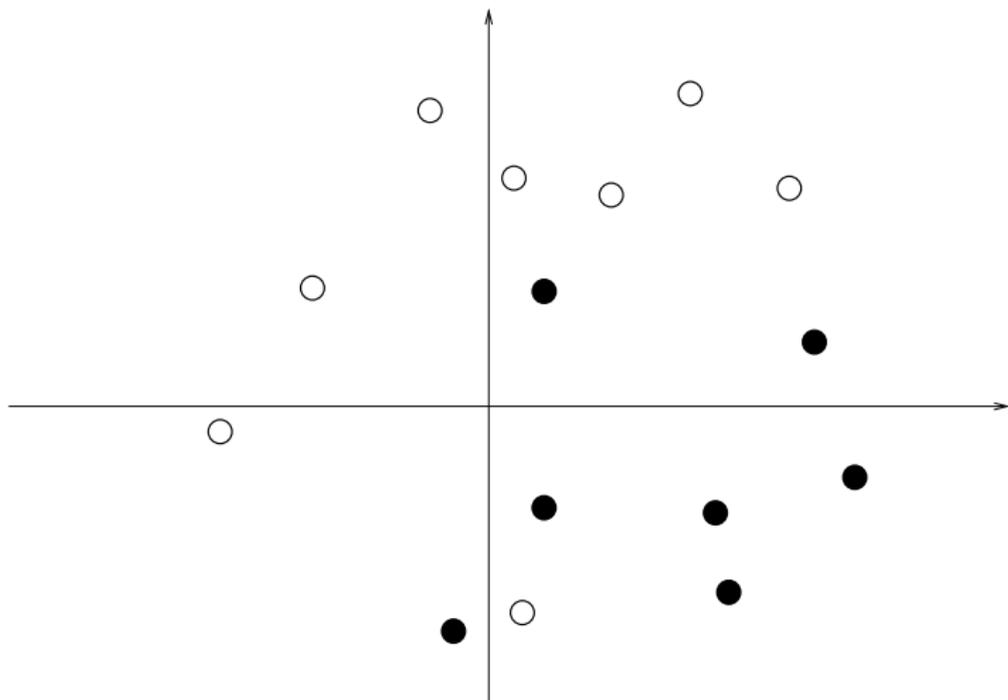
$$\begin{cases} \alpha_i [y_i f(\mathbf{x}_i) + \xi_i - 1] = 0, \\ \left(\alpha_i - \frac{y_i}{2\lambda_n}\right) \xi_i = 0. \end{cases}$$

## Analysis of KKT conditions

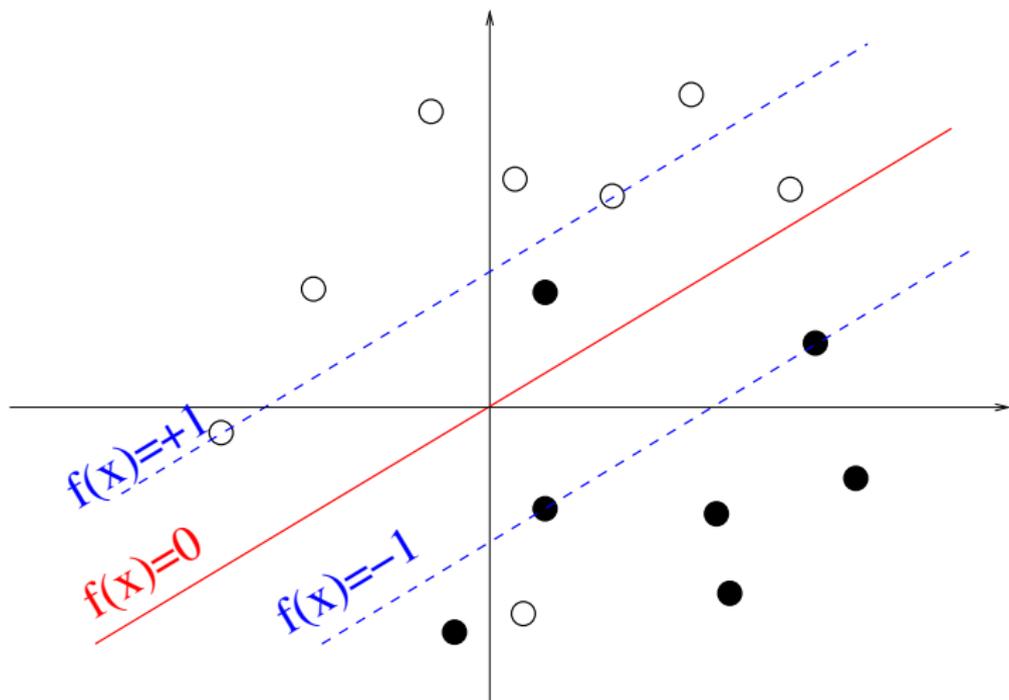
$$\begin{cases} \alpha_i [y_i f(\mathbf{x}_i) + \xi_i - 1] = 0, \\ (\alpha_i - \frac{y_i}{2\lambda_n}) \xi_i = 0. \end{cases}$$

- If  $\alpha_i = 0$ , then the second constraint is active:  $\xi_i = 0$ . This implies  $y_i f(\mathbf{x}_i) \geq 1$ .
- If  $0 < \mathbf{y}_i \alpha_i < \frac{1}{2\lambda_n}$ , then both constraints are active:  $\xi_i = 0$  et  $y_i f(\mathbf{x}_i) + \xi_i - 1 = 0$ . This implies  $y_i f(\mathbf{x}_i) = 1$ .
- If  $\alpha_i = \frac{y_i}{2\lambda_n}$ , then the second constraint is not active ( $\xi_i \geq 0$ ) while the first one is active:  $y_i f(\mathbf{x}_i) + \xi_i = 1$ . This implies  $y_i f(\mathbf{x}_i) \leq 1$

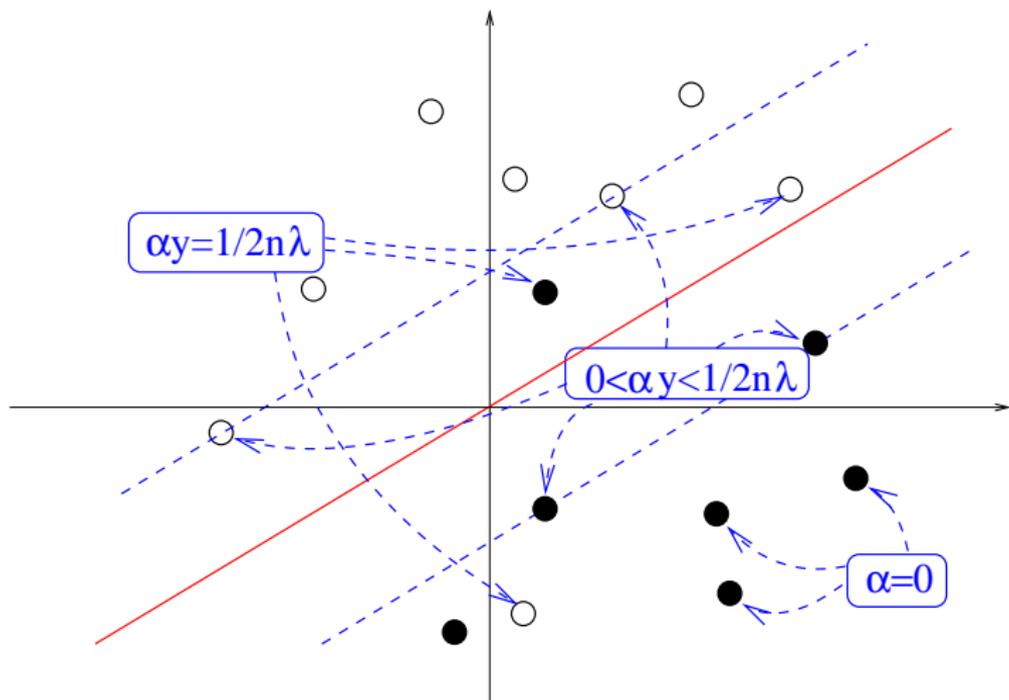
## Geometric interpretation



# Geometric interpretation



# Geometric interpretation



# Support vectors

## Consequence of KKT conditions

- The training points with  $\alpha_i \neq 0$  are called **support vectors**.
- Only support vectors are important for the classification of new points:

$$\forall \mathbf{x} \in \mathcal{X}, \quad f(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}) = \sum_{i \in SV} \alpha_i K(\mathbf{x}_i, \mathbf{x}),$$

where  $SV$  is the set of support vectors.

## Consequences

- The solution is **sparse** in  $\alpha$ , leading to **fast algorithms** for training (use of decomposition methods).
- The **classification** of a new point only involves kernel evaluations with support vectors (fast).

## Remark: C-SVM

- Often the SVM optimization problem is written in terms of a regularization parameter  $C$  instead of  $\lambda$  as follows:

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

- This is equivalent to our formulation with  $C = \frac{1}{2n\lambda}$ .
- The SVM optimization problem is then:

$$\max_{\alpha \in \mathbb{R}^d} 2 \sum_{i=1}^n \alpha_i \mathbf{y}_i - \sum_{i,j=1}^n \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j),$$

subject to:

$$0 \leq y_i \alpha_i \leq C, \quad \text{for } i = 1, \dots, n.$$

- This formulation is often called **C-SVM**.

## Remark: 2-SVM

- A variant of the SVM, sometimes called 2-SVM, is obtained by replacing the hinge loss by the square hinge loss:

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

- After some computation (left as exercise) we find that the dual problem of the 2-SVM is:

$$\max_{\alpha \in \mathbb{R}^d} 2\alpha^\top \mathbf{y} - \alpha^\top (\mathbf{K} + n\lambda I) \alpha,$$

subject to:

$$0 \leq y_i \alpha_i, \quad \text{for } i = 1, \dots, n.$$

- This is therefore **equivalent** to the previous SVM with the kernel  $\mathbf{K} + n\lambda I$  and  $C = +\infty$

# Kernels Methods

## Unsupervised Learning

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
  - Kernel K-means and spectral clustering
  - Kernel PCA
  - A quick note on kernel CCA
- 4 The Kernel Jungle

# The K-means algorithm

K-means is probably the most popular algorithm for **clustering**.

## Optimization point of view

Given data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$  in  $\mathbb{R}^p$ , it consists of performing alternate minimization steps for optimizing the following cost function

$$\min_{\substack{\boldsymbol{\mu}_j \in \mathbb{R}^p \text{ for } j=1, \dots, k \\ s_i \in \{1, \dots, k\}, \text{ for } i=1, \dots, n}} \sum_{i=1}^n \|\mathbf{x}_i - \boldsymbol{\mu}_{s_i}\|_2^2.$$

K-means alternates between two steps:

### 1 cluster assignment:

Given fixed  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$ , assign each  $\mathbf{x}_i$  to its closest centroid

$$\forall i, \quad s_i \in \operatorname{argmin}_{s \in \{1, \dots, k\}} \|\mathbf{x}_i - \boldsymbol{\mu}_s\|_2^2.$$

# The K-means algorithm

K-means is probably the most popular algorithm for **clustering**.

## Optimization point of view

Given data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$  in  $\mathbb{R}^p$ , it consists of performing alternate minimization steps for optimizing the following cost function

$$\min_{\substack{\mu_j \in \mathbb{R}^p \\ s_i \in \{1, \dots, k\}}} \sum_{i=1}^n \|\mathbf{x}_i - \mu_{s_i}\|_2^2 \quad \text{for } j=1, \dots, k \text{ and } i=1, \dots, n$$

K-means alternates between two steps:

### 2 centroids update:

Given the previous assignments  $s_1, \dots, s_n$ , update the centroids

$$\forall j, \quad \mu_j = \operatorname{argmin}_{\mu \in \mathbb{R}^p} \sum_{i: s_i=j} \|\mathbf{x}_i - \mu\|_2^2.$$

# The K-means algorithm

K-means is probably the most popular algorithm for **clustering**.

## Optimization point of view

Given data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$  in  $\mathbb{R}^p$ , it consists of performing alternate minimization steps for optimizing the following cost function

$$\min_{\substack{\boldsymbol{\mu}_j \in \mathbb{R}^p \\ s_j \in \{1, \dots, k\}}} \sum_{i=1}^n \|\mathbf{x}_i - \boldsymbol{\mu}_{s_i}\|_2^2$$

K-means alternates between two steps:

### 2 centroids update:

Given the previous assignments  $s_1, \dots, s_n$ , update the centroids

$$\Leftrightarrow \forall j, \quad \boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{i:s_i=j} \mathbf{x}_i.$$

## Kernel K-means and spectral clustering

We may now modify the objective to **operate in a RKHS**. Given data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$  in  $\mathcal{X}$  and a p.d. kernel  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  with  $\mathcal{H}$  its RKHS, the new objective becomes

$$\min_{\substack{\boldsymbol{\mu}_j \in \mathcal{H} \\ s_j \in \{1, \dots, k\}}} \sum_{i=1}^n \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_{s_i}\|_{\mathcal{H}}^2.$$

To optimize the cost function, we will first use the following Proposition

### Proposition

The center of mass  $\varphi_n = \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{x}_i)$  solves the following optimization problem

$$\min_{\boldsymbol{\mu} \in \mathcal{H}} \sum_{i=1}^n \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}\|_{\mathcal{H}}^2.$$

# Kernel K-means and spectral clustering

## Proof

$$\begin{aligned}\frac{1}{n} \sum_{i=1}^n \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}\|_{\mathcal{H}}^2 &= \frac{1}{n} \sum_{i=1}^n \|\varphi(\mathbf{x}_i)\|_{\mathcal{H}}^2 - \left\langle \frac{2}{n} \sum_{i=1}^n \varphi(\mathbf{x}_i), \boldsymbol{\mu} \right\rangle_{\mathcal{H}} + \|\boldsymbol{\mu}\|_{\mathcal{H}}^2 \\ &= \frac{1}{n} \sum_{i=1}^n \|\varphi(\mathbf{x}_i)\|_{\mathcal{H}}^2 - 2 \langle \varphi_n, \boldsymbol{\mu} \rangle_{\mathcal{H}} + \|\boldsymbol{\mu}\|_{\mathcal{H}}^2 \\ &= \frac{1}{n} \sum_{i=1}^n \|\varphi(\mathbf{x}_i)\|_{\mathcal{H}}^2 - \|\varphi_n\|_{\mathcal{H}}^2 + \|\varphi_n - \boldsymbol{\mu}\|_{\mathcal{H}}^2,\end{aligned}$$

which is minimum for  $\boldsymbol{\mu} = \varphi_n$ .

## Kernel K-means and spectral clustering

Back with the objective,

$$\min_{\substack{\mu_j \in \mathcal{H} \text{ for } j=1, \dots, k \\ s_i \in \{1, \dots, k\} \text{ for } i=1, \dots, n}} \sum_{i=1}^n \|\varphi(\mathbf{x}_i) - \mu_{s_i}\|_{\mathcal{H}}^2,$$

we know that given assignments  $s_i$ , the optimal  $\mu_j$  are the centers of mass of the respective clusters and we obtain the equivalent objective:

$$\min_{\substack{s_i \in \{1, \dots, k\} \\ \text{for } i=1, \dots, n}} \sum_{i=1}^n \left\| \varphi(\mathbf{x}_i) - \frac{1}{|C_{s_i}|} \sum_{j \in C_{s_i}} \varphi(\mathbf{x}_j) \right\|_{\mathcal{H}}^2,$$

or, after short calculations,

$$\min_{\substack{s_i \in \{1, \dots, k\} \\ \text{for } i=1, \dots, n}} \sum_{i=1}^n K(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{|C_{s_i}|} \sum_{j \in C_{s_i}} K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{|C_{s_i}|^2} \sum_{j \in C_{s_i}} \sum_{l \in C_{s_i}} K(\mathbf{x}_j, \mathbf{x}_l).$$

## Kernel K-means and spectral clustering

and, after removing the constant terms, we obtain the objective

$$\min_{\substack{s_i \in \{1, \dots, k\} \\ \text{for } i=1, \dots, n}} \sum_{i=1}^n -\frac{1}{|C_{s_i}|} \sum_{j \in C_{s_i}} K(\mathbf{x}_i, \mathbf{x}_j), \quad (\star)$$

The objective can be expressed with **pairwise kernel comparisons**. Unfortunately, the problem is hard and we need an appropriate strategy to obtain an **approximate solution**.

### Greedy approach: kernel K-means

At every iteration,

- Update the sets  $C_l$ ,  $l = 1, \dots, k$  given current assignments  $s_i$ 's.
- Update the assignments by minimizing  $(\star)$  keeping the sets  $C_l$  fixed.

The algorithm is similar to the traditional K-means algorithm.

## Kernel K-means and spectral clustering

Another approach consists of **relaxing the non-convex problem with a feasible one**, which yields a class of algorithms called **spectral clustering**. First, we rewrite the objective function as

$$\min_{\substack{s_i \in \{1, \dots, k\} \\ \text{for } i=1, \dots, n}} \sum_{l=1}^k \sum_{i, j \in C_l} -\frac{1}{|C_l|} K(\mathbf{x}_i, \mathbf{x}_j).$$

and we introduce

- the binary matrix  $\mathbf{A}$  in  $\{0, 1\}^{n \times k}$  such that  $[\mathbf{A}]_{ij} = 1$  if  $s_i = j$  and 0 otherwise.
- a diagonal matrix  $\mathbf{D}$  in  $\mathbb{R}^{l \times l}$  with diagonal entries  $[\mathbf{D}]_{jj}$  equal to the inverse of the number of elements in cluster  $j$ .

and the objective can be rewritten (proof is easy and left as an exercise)

$$\min_{\mathbf{A}, \mathbf{D}} \left[ -\text{trace}(\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{K} \mathbf{A} \mathbf{D}^{1/2}) \right].$$

## Kernel K-means and spectral clustering

$$\min_{\mathbf{A}, \mathbf{D}} \left[ -\text{trace}(\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{K} \mathbf{A} \mathbf{D}^{1/2}) \right].$$

The constraints on  $\mathbf{A}, \mathbf{D}$  are such that  $\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{A} \mathbf{D}^{1/2} = \mathbf{I}$  (exercise). A natural relaxation consists of dropping the constraints on  $\mathbf{A}$  and instead optimize over  $\mathbf{Z} = \mathbf{A} \mathbf{D}^{1/2}$ :

$$\max_{\mathbf{Z} \in \mathbb{R}^{n \times k}} \text{trace}(\mathbf{Z}^\top \mathbf{K} \mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}.$$

A solution  $\mathbf{Z}^*$  to this problem may be obtained by computing the eigenvectors of  $\mathbf{K}$  associated to the  $k$ -largest eigenvalues. As we will see in a few slides, this procedure is related to the **kernel PCA** algorithm.

## Kernel K-means and spectral clustering

$$\min_{\mathbf{A}, \mathbf{D}} \left[ -\text{trace}(\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{K} \mathbf{A} \mathbf{D}^{1/2}) \right].$$

The constraints on  $\mathbf{A}, \mathbf{D}$  are such that  $\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{A} \mathbf{D}^{1/2} = \mathbf{I}$  (exercise). A natural relaxation consists of dropping the constraints on  $\mathbf{A}$  and instead optimize over  $\mathbf{Z} = \mathbf{A} \mathbf{D}^{1/2}$ :

$$\max_{\mathbf{Z} \in \mathbb{R}^{n \times k}} \text{trace}(\mathbf{Z}^\top \mathbf{K} \mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}.$$

A solution  $\mathbf{Z}^*$  to this problem may be obtained by computing the eigenvectors of  $\mathbf{K}$  associated to the  $k$ -largest eigenvalues. As we will see in a few slides, this procedure is related to the **kernel PCA** algorithm.

### Question

How do we obtain an approximate solution  $(\mathbf{A}, \mathbf{D})$  of the original problem from  $\mathbf{Z}^*$ ?

## Kernel K-means and spectral clustering

$$\min_{\mathbf{A}, \mathbf{D}} \left[ -\text{trace}(\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{K} \mathbf{A} \mathbf{D}^{1/2}) \right].$$

The constraints on  $\mathbf{A}, \mathbf{D}$  are such that  $\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{A} \mathbf{D}^{1/2} = \mathbf{I}$  (exercise). A natural relaxation consists of dropping the constraints on  $\mathbf{A}$  and instead optimize over  $\mathbf{Z} = \mathbf{A} \mathbf{D}^{1/2}$ :

$$\max_{\mathbf{Z} \in \mathbb{R}^{n \times k}} \text{trace}(\mathbf{Z}^\top \mathbf{K} \mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}.$$

A solution  $\mathbf{Z}^*$  to this problem may be obtained by computing the eigenvectors of  $\mathbf{K}$  associated to the  $k$ -largest eigenvalues. As we will see in a few slides, this procedure is related to the **kernel PCA** algorithm.

### Answer 1

With the original constraints on  $\mathbf{A}$ , every row of  $\mathbf{A}$  has a single non-zero entry  $\Rightarrow$  compute the maximum entry of every row of  $\mathbf{Z}^*$ .

## Kernel K-means and spectral clustering

$$\min_{\mathbf{A}, \mathbf{D}} \left[ -\text{trace}(\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{K} \mathbf{A} \mathbf{D}^{1/2}) \right].$$

The constraints on  $\mathbf{A}, \mathbf{D}$  are such that  $\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{A} \mathbf{D}^{1/2} = \mathbf{I}$  (exercise). A natural relaxation consists of dropping the constraints on  $\mathbf{A}$  and instead optimize over  $\mathbf{Z} = \mathbf{A} \mathbf{D}^{1/2}$ :

$$\max_{\mathbf{Z} \in \mathbb{R}^{n \times k}} \text{trace}(\mathbf{Z}^\top \mathbf{K} \mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}.$$

A solution  $\mathbf{Z}^*$  to this problem may be obtained by computing the eigenvectors of  $\mathbf{K}$  associated to the  $k$ -largest eigenvalues. As we will see in a few slides, this procedure is related to the **kernel PCA** algorithm.

### Answer 2

Normalize the rows of  $\mathbf{Z}^*$  to have unit  $\ell_2$ -norm, and apply the traditional K-means algorithm on the rows. This is called **spectral clustering**.

## Kernel K-means and spectral clustering

$$\min_{\mathbf{A}, \mathbf{D}} \left[ -\text{trace}(\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{K} \mathbf{A} \mathbf{D}^{1/2}) \right].$$

The constraints on  $\mathbf{A}, \mathbf{D}$  are such that  $\mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{A} \mathbf{D}^{1/2} = \mathbf{I}$  (exercise). A natural relaxation consists of dropping the constraints on  $\mathbf{A}$  and instead optimize over  $\mathbf{Z} = \mathbf{A} \mathbf{D}^{1/2}$ :

$$\max_{\mathbf{Z} \in \mathbb{R}^{n \times k}} \text{trace}(\mathbf{Z}^\top \mathbf{K} \mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}.$$

A solution  $\mathbf{Z}^*$  to this problem may be obtained by computing the eigenvectors of  $\mathbf{K}$  associated to the  $k$ -largest eigenvalues. As we will see in a few slides, this procedure is related to the **kernel PCA** algorithm.

### Answer 3

Choose another variant of the previous procedures.

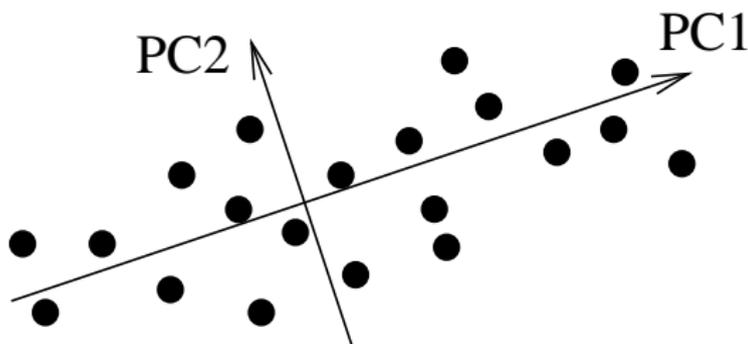
# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
  - Kernel K-means and spectral clustering
  - **Kernel PCA**
  - A quick note on kernel CCA
- 4 The Kernel Jungle

# Principal Component Analysis (PCA)

## Classical setting

- Let  $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  be a set of vectors ( $\mathbf{x}_i \in \mathbb{R}^d$ )
- PCA is a classical algorithm in multivariate statistics to define a set of orthogonal directions that capture the maximum variance
- Applications: low-dimensional representation of high-dimensional points, visualization



# Principal Component Analysis (PCA)

## Formalization

- Assume that the data are **centered** (otherwise center them as preprocessing), i.e.:

$$\sum_{i=1}^n \mathbf{x}_i = 0.$$

- The **orthogonal projection** onto a direction  $\mathbf{w} \in \mathbb{R}^d$  is the function  $h_{\mathbf{w}} : \mathcal{X} \rightarrow \mathbb{R}$  defined by:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^{\top} \frac{\mathbf{w}}{\|\mathbf{w}\|}.$$

# Principal Component Analysis (PCA)

## Formalization

- The **empirical variance** captured by  $h_{\mathbf{w}}$  is:

$$\hat{\text{var}}(h_{\mathbf{w}}) := \frac{1}{n} \sum_{i=1}^n h_{\mathbf{w}}(\mathbf{x}_i)^2 = \frac{1}{n} \sum_{i=1}^n \frac{(\mathbf{x}_i^{\top} \mathbf{w})^2}{\|\mathbf{w}\|^2}.$$

- The  $i$ -th principal direction  $\mathbf{w}_i$  ( $i = 1, \dots, d$ ) is defined by:

$$\mathbf{w}_i = \underset{\mathbf{w} \perp \{\mathbf{w}_1, \dots, \mathbf{w}_{i-1}\}}{\text{arg max}} \hat{\text{var}}(h_{\mathbf{w}}).$$

# Principal Component Analysis (PCA)

## Solution

- Let  $\mathbf{X}$  be the  $n \times d$  data matrix whose rows are the vectors  $\mathbf{x}_1, \dots, \mathbf{x}_n$ . We can then write:

$$\hat{\text{var}}(h_{\mathbf{w}}) = \frac{1}{n} \sum_{i=1}^n \frac{(\mathbf{x}_i^{\top} \mathbf{w})^2}{\|\mathbf{w}\|^2} = \frac{1}{n} \frac{\mathbf{w}^{\top} \mathbf{X}^{\top} \mathbf{X} \mathbf{w}}{\mathbf{w}^{\top} \mathbf{w}}.$$

- The solutions of:

$$\mathbf{w}_i = \arg \max_{\mathbf{w} \perp \{\mathbf{w}_1, \dots, \mathbf{w}_{i-1}\}} \frac{1}{n} \frac{\mathbf{w}^{\top} \mathbf{X}^{\top} \mathbf{X} \mathbf{w}}{\mathbf{w}^{\top} \mathbf{w}}$$

are the **successive eigenvectors of  $\mathbf{K} = \mathbf{X}^{\top} \mathbf{X}$** , ranked by decreasing eigenvalues.

## Functional point of view

- Let  $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$  be the linear kernel.
- The associated RKHS  $\mathcal{H}$  is the set of linear functions:

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x},$$

endowed with the norm  $\|f_{\mathbf{w}}\|_{\mathcal{H}} = \|\mathbf{w}\|_{\mathbb{R}^d}$ .

- Therefore we can write:

$$\hat{\text{var}}(h_{\mathbf{w}}) = \frac{1}{n} \sum_{i=1}^n \frac{(\mathbf{x}_i^\top \mathbf{w})^2}{\|\mathbf{w}\|^2} = \frac{1}{n \|\mathbf{w}\|^2} \sum_{i=1}^n f_{\mathbf{w}}(\mathbf{x}_i)^2.$$

- Moreover,  $\mathbf{w} \perp \mathbf{w}' \Leftrightarrow f_{\mathbf{w}} \perp f_{\mathbf{w}'}$ .

## Functional point of view

- In other words, PCA solves, for  $i = 1, \dots, d$ :

$$f_i = \arg \max_{f \perp \{f_1, \dots, f_{i-1}\}} \frac{1}{n \|f\|^2} \sum_{i=1}^n f(\mathbf{x}_i)^2.$$

- We can apply the representer theorem (*exercice: check that is is also valid in a linear subspace*): for  $i = 1, \dots, d$ , we have:

$$\forall \mathbf{x} \in \mathcal{X}, \quad f_i(\mathbf{x}) = \sum_{j=1}^n \alpha_{i,j} K(\mathbf{x}_j, \mathbf{x}),$$

with  $\alpha_i = (\alpha_{i,1}, \dots, \alpha_{i,n})^\top \in \mathbb{R}^n$ .

## Functional point of view

- Therefore we have:

$$\|f_i\|_{\mathcal{H}}^2 = \sum_{k,l=1}^d \alpha_{i,k} \alpha_{i,l} K(\mathbf{x}_k, \mathbf{x}_l) = \boldsymbol{\alpha}_i^\top \mathbf{K} \boldsymbol{\alpha}_i,$$

- Similarly:

$$\sum_{k=1}^n f_i(\mathbf{x}_k)^2 = \boldsymbol{\alpha}_i^\top \mathbf{K}^2 \boldsymbol{\alpha}_i.$$

## Functional point of view

PCA maximizes in  $\alpha$  the function:

$$\alpha_j = \arg \max_{\alpha} \frac{\alpha^\top \mathbf{K}^2 \alpha}{n \alpha^\top \mathbf{K} \alpha},$$

under the constraints:

$$\alpha_j^\top \mathbf{K} \alpha_j = 0 \quad \text{for } j = 1, \dots, i-1.$$

## Solution

- Let  $(\mathbf{u}_1, \dots, \mathbf{u}_n)$  be an orthonormal basis of eigenvectors of  $K$  with eigenvalues  $\lambda_1 \geq \dots \geq \lambda_n \geq 0$ .
- Let  $\alpha_i = \sum_{j=1}^n \beta_{ij} \mathbf{u}_j$ , then

$$\frac{\alpha_i^\top \mathbf{K}^2 \alpha_i}{n \alpha_i^\top \mathbf{K} \alpha_i} = \frac{\sum_{j=1}^n \beta_{ij}^2 \lambda_j^2}{n \sum_{j=1}^n \beta_{ij}^2 \lambda_j},$$

which is maximized at  $\alpha_1 = \beta_{11} \mathbf{u}_1$ ,  $\alpha_2 = \beta_{22} \mathbf{u}_2$ , etc...

# Normalization

- For  $\alpha_j = \beta_{jj}\mathbf{u}_j$ , we want:

$$1 = \|f_j\|_{\mathcal{H}}^2 = \alpha_j^\top \mathbf{K} \alpha_j = \beta_{jj}^2 \lambda_j.$$

- Therefore:

$$\alpha_j = \frac{1}{\sqrt{\lambda_j}} \mathbf{u}_j.$$

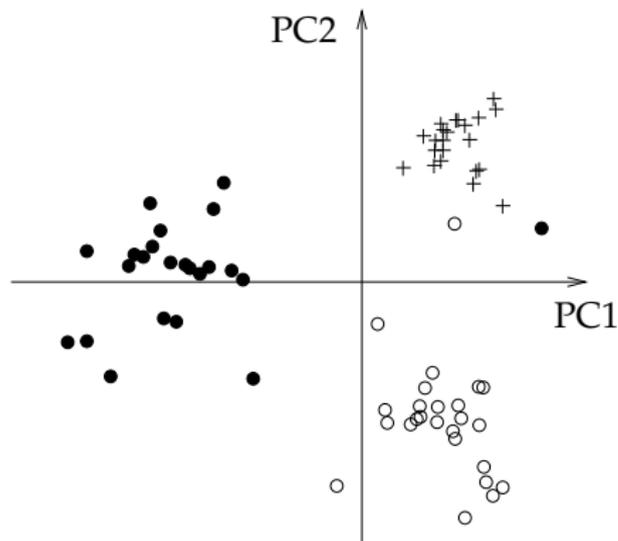
## Kernel PCA: summary

- 1 Center the Gram matrix
- 2 Compute the first eigenvectors  $(\mathbf{u}_i, \lambda_i)$
- 3 Normalize the eigenvectors  $\alpha_i = \mathbf{u}_i / \sqrt{\lambda_i}$
- 4 The projections of the points onto the  $i$ -th eigenvector is given by  $\mathbf{K}\alpha_i$

## Kernel PCA: remarks

- In this formulation, we must **diagonalize the centered kernel Gram matrix**, instead of the covariance matrix in the classical setting
- *Exercice: check that  $\mathbf{X}^\top \mathbf{X}$  and  $\mathbf{X}\mathbf{X}^\top$  have the same spectrum (up to 0 eigenvalues) and that the eigenvectors are related by a simple relationship.*
- This formulation remains valid for any p.d. kernel: this is **kernel PCA**
- **Applications:** nonlinear PCA with nonlinear kernels for vectors, PCA of non-vector objects (strings, graphs..) with specific kernels...

## Example



A set of 74 human tRNA sequences is analyzed using a kernel for sequences (the second-order marginalized kernel based on SCFG). This set of tRNAs contains three classes, called Ala-AGC (*white circles*), Asn-GTT (*black circles*) and Cys-GCA (*plus symbols*) (from Tsuda et al., 2003).

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
  - Kernel K-means and spectral clustering
  - Kernel PCA
  - A quick note on kernel CCA
- 4 The Kernel Jungle

# Canonical Correlation Analysis (CCA)

Given two views  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$  in  $\mathbb{R}^{p \times n}$  and  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n]$  in  $\mathbb{R}^{d \times n}$  of the same dataset, the goal of canonical correlation analysis (CCA) is to find **pairs of directions** in the two views that are **maximally correlated**.

## Formulation

Assuming that the datasets are centered, we want to maximize

$$\max_{\mathbf{w}_a \in \mathbb{R}^p, \mathbf{w}_b \in \mathbb{R}^d} \frac{\frac{1}{n} \sum_{i=1}^n \mathbf{w}_a^\top \mathbf{x}_i \mathbf{y}_i^\top \mathbf{w}_b}{\left(\frac{1}{n} \sum_{i=1}^n \mathbf{w}_a^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{w}_a\right)^{1/2} \left(\frac{1}{n} \sum_{i=1}^n \mathbf{w}_b^\top \mathbf{y}_i \mathbf{y}_i^\top \mathbf{w}_b\right)^{1/2}}.$$

Assuming that the pairs  $(\mathbf{x}_i, \mathbf{y}_i)$  are i.i.d. samples from an unknown distribution, CCA seeks to maximize

$$\max_{\mathbf{w}_a \in \mathbb{R}^p, \mathbf{w}_b \in \mathbb{R}^d} \frac{\text{cov}(\mathbf{w}_a^\top X, \mathbf{w}_b^\top Y)}{\sqrt{\text{var}(\mathbf{w}_a^\top X)} \sqrt{\text{var}(\mathbf{w}_b^\top Y)}}.$$

# Canonical Correlation Analysis (CCA)

Given two views  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$  in  $\mathbb{R}^{p \times n}$  and  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n]$  in  $\mathbb{R}^{d \times n}$  of the same dataset, the goal of canonical correlation analysis (CCA) is to find **pairs of directions** in the two views that are **maximally correlated**.

## Formulation

Assuming that the datasets are centered, we want to maximize

$$\max_{\mathbf{w}_a \in \mathbb{R}^p, \mathbf{w}_b \in \mathbb{R}^d} \frac{\frac{1}{n} \sum_{i=1}^n \mathbf{w}_a^\top \mathbf{x}_i \mathbf{y}_i^\top \mathbf{w}_b}{\left(\frac{1}{n} \sum_{i=1}^n \mathbf{w}_a^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{w}_a\right)^{1/2} \left(\frac{1}{n} \sum_{i=1}^n \mathbf{w}_b^\top \mathbf{y}_i \mathbf{y}_i^\top \mathbf{w}_b\right)^{1/2}}.$$

It is possible to show that this is an **generalized eigenvalue problem** (exercise, or see Section 6.5 of Shawe-Taylor and Cristianini 2004).

The above problem provides the **first pair of canonical directions**. Next directions can be obtained by solving the same problem under the constraint that they are **orthogonal to the previous canonical directions**.

# Kernel Canonical Correlation Analysis

Similar to kernel PCA, **it is possible to operate in a RKHS**. Given two p.d. kernels  $K_a, K_b : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ , we can obtain two “views” of a dataset  $\mathbf{x}_1, \dots, \mathbf{x}_n$  in  $\mathcal{X}^n$ :

$$(\varphi_a(\mathbf{x}_1), \dots, \varphi_a(\mathbf{x}_n)) \quad \text{and} \quad (\varphi_b(\mathbf{x}_1), \dots, \varphi_b(\mathbf{x}_n)),$$

where  $\varphi_a : \mathcal{X} \rightarrow \mathcal{H}_a$  and  $\varphi_b : \mathcal{X} \rightarrow \mathcal{H}_b$  are the embeddings in the RKHSs  $\mathcal{H}_a$  of  $K_a$  and  $\mathcal{H}_b$  of  $K_b$ , respectively. Then, we may formulate **kernel CCA** as the following optimization problem

$$\max_{f_a \in \mathcal{H}_a, f_b \in \mathcal{H}_b} \frac{\frac{1}{n} \sum_{i=1}^n \langle f_a, \varphi_a(\mathbf{x}_i) \rangle_{\mathcal{H}_a} \langle \varphi_b(\mathbf{x}_i), f_b \rangle_{\mathcal{H}_b}}{\left( \frac{1}{n} \sum_{i=1}^n \langle f_a, \varphi_a(\mathbf{x}_i) \rangle_{\mathcal{H}_a}^2 \right)^{1/2} \left( \frac{1}{n} \sum_{i=1}^n \langle \varphi_b(\mathbf{x}_i), f_b \rangle_{\mathcal{H}_b}^2 \right)^{1/2}}.$$

# Kernel Canonical Correlation Analysis

Similar to kernel PCA, **it is possible to operate in a RKHS**. Given two p.d. kernels  $K_a, K_b : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ , we can obtain two “views” of a dataset  $\mathbf{x}_1, \dots, \mathbf{x}_n$  in  $\mathcal{X}^n$ :

$$(\varphi_a(\mathbf{x}_1), \dots, \varphi_a(\mathbf{x}_n)) \quad \text{and} \quad (\varphi_b(\mathbf{x}_1), \dots, \varphi_b(\mathbf{x}_n)),$$

where  $\varphi_a : \mathcal{X} \rightarrow \mathcal{H}_a$  and  $\varphi_b : \mathcal{X} \rightarrow \mathcal{H}_b$  are the embeddings in the RKHSs  $\mathcal{H}_a$  of  $K_a$  and  $\mathcal{H}_b$  of  $K_b$ , respectively. Then, we may formulate **kernel CCA** as the following optimization problem

$$\max_{f_a \in \mathcal{H}_a, f_b \in \mathcal{H}_b} \frac{\frac{1}{n} \sum_{i=1}^n f_a(\mathbf{x}_i) f_b(\mathbf{x}_i)}{\left(\frac{1}{n} \sum_{i=1}^n f_a(\mathbf{x}_i)^2\right)^{1/2} \left(\frac{1}{n} \sum_{i=1}^n f_b(\mathbf{x}_i)^2\right)^{1/2}}.$$

# Kernel Canonical Correlation Analysis

Up to a few technical details (exercise), we can apply the **representer theorem** and look for solutions  $f_a(\cdot) = \sum_{i=1}^n \alpha_i K_a(\mathbf{x}_i, \cdot)$  and  $f_b(\cdot) = \sum_{i=1}^n \beta_i K_b(\mathbf{x}_i, \cdot)$ . We finally obtain the formulation

$$\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \frac{\frac{1}{n} \sum_{i=1}^n [\mathbf{K}_a \alpha]_i [\mathbf{K}_b \beta]_i}{\left(\frac{1}{n} \sum_{i=1}^n [\mathbf{K}_a \alpha]_i^2\right)^{1/2} \left(\frac{1}{n} \sum_{i=1}^n [\mathbf{K}_b \beta]_i^2\right)^{1/2}},$$

which is equivalent to

$$\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \frac{\alpha^\top \mathbf{K}_a \mathbf{K}_b \beta}{\left(\alpha^\top \mathbf{K}_a^2 \alpha\right)^{1/2} \left(\beta^\top \mathbf{K}_b^2 \beta\right)^{1/2}},$$

or, after removing the scaling ambiguity for  $\alpha$  and  $\beta$ ,

$$\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \alpha^\top \mathbf{K}_a \mathbf{K}_b \beta \quad \text{s.t.} \quad \alpha^\top \mathbf{K}_a^2 \alpha = 1 \quad \text{and} \quad \beta^\top \mathbf{K}_b^2 \beta = 1.$$

# Kernel Canonical Correlation Analysis

## Remarks

- kernel CCA also yields a generalized eigenvalue problem.
- the subsequent canonical directions are obtained by solving the same problem with additional orthogonality constraints.

# The Kernel Jungle

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - Kernels on graphs

# Motivations

- The RKHS norm is related to the **smoothness** of functions.
- Smoothness of a function is naturally quantified by **Sobolev norms** (in particular  $L_2$  norms of derivatives), or by the decay of the Fourier transform.
- In this section, we introduce several kernels where this link is explicit, and we make a general link between RKHS and **Green functions defined by differential operators**.

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
    - Shift-invariant kernels
      - Generalization to semigroups
      - Mercer kernels
      - RKHS and Green functions
    - Kernels for generative models
    - Kernels for biological sequences
    - Kernels for graphs
    - Kernels on graphs

# Translation invariant kernels

## Definition

A kernel  $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$  is called **translation invariant** (t.i.) if it only depends on the difference between its argument, i.e.:

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2d}, \quad K(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x} - \mathbf{y}).$$

## Examples

- Gaussian kernel (or RBF kernel)

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{y}\|_2^2}.$$

- Laplace kernel

$$K(\mathbf{x}, \mathbf{y}) = e^{-\alpha \|\mathbf{x} - \mathbf{y}\|_1}.$$

## In case of...

### Definition

Let  $f \in L^1(\mathbb{R}^d)$ . The **Fourier transform** of  $f$ , denoted  $\hat{f}$  or  $\mathcal{F}[f]$ , is the function defined for all  $\omega \in \mathbb{R}^d$  by:

$$\hat{f}(\omega) = \int_{\mathbb{R}^d} e^{-ix \cdot \omega} f(\mathbf{x}) d\mathbf{x}.$$

## In case of...

### Properties

- $\hat{f}$  is complex-valued, continuous, tends to 0 at infinity and  $\|\hat{f}\|_{L^\infty} \leq \|f\|_{L^1}$ .
- If  $\hat{f} \in L^1(\mathbb{R}^d)$ , then the **inverse Fourier formula** holds:

$$\forall \mathbf{x} \in \mathbb{R}^d, \quad f(\mathbf{x}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} e^{i\mathbf{x} \cdot \boldsymbol{\omega}} \hat{f}(\boldsymbol{\omega}) d\boldsymbol{\omega}.$$

- If  $f \in L^1(\mathbb{R}^d)$  is square integrable, then **Parseval's formula** holds:

$$\int_{\mathbb{R}^d} |f(\mathbf{x})|^2 d\mathbf{x} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\hat{f}(\boldsymbol{\omega})|^2 d\boldsymbol{\omega}.$$

# Translation invariant kernels

## Definition

A kernel  $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$  is called **translation invariant** (t.i.) if it only depends on the difference between its argument, i.e.:

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2d}, \quad K(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x} - \mathbf{y}).$$

## Intuition

If  $K$  is t.i. and  $\kappa \in L^1(\mathbb{R}^d)$ , then

$$\begin{aligned} \kappa(\mathbf{x} - \mathbf{y}) &= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} e^{i(\mathbf{x}-\mathbf{y}) \cdot \boldsymbol{\omega}} \hat{\kappa}(\boldsymbol{\omega}) d\boldsymbol{\omega} \\ &= \int_{\mathbb{R}^d} \frac{\hat{\kappa}(\boldsymbol{\omega})}{(2\pi)^d} e^{i\boldsymbol{\omega} \cdot \mathbf{x}} e^{-i\boldsymbol{\omega} \cdot \mathbf{y}} d\boldsymbol{\omega}. \end{aligned}$$

# RKHS of translation invariant kernels

## Theorem

Let  $K$  be a translation invariant p.d. kernel, such that  $\kappa$  is integrable on  $\mathbb{R}^d$  as well as its Fourier transform  $\hat{\kappa}$ . The subset  $\mathcal{H}_K$  of  $L_2(\mathbb{R}^d)$  that consists of integrable and continuous functions  $f$  such that:

$$\|f\|_K^2 := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \frac{|\hat{f}(\omega)|^2}{\hat{\kappa}(\omega)} d\omega < +\infty,$$

endowed with the inner product:

$$\langle f, g \rangle := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \frac{\hat{f}(\omega) \hat{g}(\omega)^*}{\hat{\kappa}(\omega)} d\omega$$

is a RKHS with  $K$  as r.k.

# Proof

For  $\mathbf{x} \in \mathbb{R}^d$ ,  $K_{\mathbf{x}}(\mathbf{y}) = K(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x} - \mathbf{y})$  therefore:

$$\hat{K}_{\mathbf{x}}(\boldsymbol{\omega}) = \int e^{-i\boldsymbol{\omega} \cdot \mathbf{u}} \kappa(\mathbf{u} - \mathbf{x}) d\mathbf{u} = e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \hat{\kappa}(\boldsymbol{\omega}).$$

This leads to  $K_{\mathbf{x}} \in \mathcal{H}$ , because:

$$\int_{\mathbb{R}^d} \frac{|\hat{K}_{\mathbf{x}}(\boldsymbol{\omega})|^2}{\hat{\kappa}(\boldsymbol{\omega})} \leq \int_{\mathbb{R}^d} |\hat{\kappa}(\boldsymbol{\omega})| < \infty,$$

Moreover, if  $f \in \mathcal{H}$  and  $\mathbf{x} \in \mathbb{R}^d$ , we have:

$$\langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \frac{\hat{K}_{\mathbf{x}}(\boldsymbol{\omega}) \hat{f}(\boldsymbol{\omega})^*}{\hat{\kappa}(\boldsymbol{\omega})} d\boldsymbol{\omega} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{f}(\boldsymbol{\omega})^* e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} = f(\mathbf{x})$$

# Application: characterization of p.d. t.i. kernels

## Theorem (Bochner)

A real-valued continuous function  $\kappa(\mathbf{x} - \mathbf{y})$  on  $\mathbb{R}^d$  is positive definite if and only if it is the Fourier-Stieltjes transform of a **symmetric, positive, and finite** Borel measure.

# Example

## Gaussian kernel

$$K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}$$

corresponds to:

$$\hat{k}(\omega) = e^{-\frac{\sigma^2\omega^2}{2}}$$

and

$$\mathcal{H} = \left\{ f : \int \left| \hat{f}(\omega) \right|^2 e^{\frac{\sigma^2\omega^2}{2}} d\omega < \infty \right\}.$$

In particular, all functions in  $\mathcal{H}$  are **infinitely differentiable** with all derivatives in  $L^2$ .

## Example

### Laplace kernel

$$K(x, y) = \frac{1}{2} e^{-\gamma|x-y|}$$

corresponds to:

$$\hat{k}(\omega) = \frac{\gamma}{\gamma^2 + \omega^2}$$

and

$$\mathcal{H} = \left\{ f : \int \left| \hat{f}(\omega) \right|^2 \frac{(\gamma^2 + \omega^2)}{\gamma} d\omega < \infty \right\},$$

the set of functions  $L^2$  differentiable with derivatives in  $L^2$  (Sobolev norm).

## Example

### Low-frequency filter

$$K(x, y) = \frac{\sin(\Omega(x - y))}{\pi(x - y)}$$

corresponds to:

$$\hat{k}(\omega) = U(\omega + \Omega) - U(\omega - \Omega)$$

and

$$\mathcal{H} = \left\{ f : \int_{|\omega| > \Omega} |\hat{f}(\omega)|^2 d\omega = 0 \right\},$$

the set of functions whose spectrum is included in  $[-\Omega, \Omega]$ .

# Outline

## 4 The Kernel Jungle

- Mercer kernels and shift-invariant kernels
  - Shift-invariant kernels
  - Generalization to semigroups
  - Mercer kernels
  - RKHS and Green functions
- Kernels for generative models
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs

# Generalization to semigroups (cf Berg et al., 1983)

## Definition

- A **semigroup**  $(S, \circ)$  is a nonempty set  $S$  equipped with an associative composition  $\circ$  and a neutral element  $e$ .
- A **semigroup with involution**  $(S, \circ, *)$  is a semigroup  $(S, \circ)$  together with a mapping  $* : S \rightarrow S$  called **involution** satisfying:
  - 1  $(s \circ t)^* = t^* \circ s^*$ , for  $s, t \in S$ .
  - 2  $(s^*)^* = s$  for  $s \in S$ .

## Examples

- Any **group**  $(G, \circ)$  is a semigroup with involution when we define  $s^* = s^{-1}$ .
- Any **abelian semigroup**  $(S, +)$  is a semigroup with involution when we define  $s^* = s$ , the **identical involution**.

# Positive definite functions on semigroups

## Definition

Let  $(S, \circ, *)$  be a semigroup with involution. A function  $\varphi : S \rightarrow \mathbb{R}$  is called **positive definite** if the function:

$$\forall s, t \in S, \quad K(s, t) = \varphi(s^* \circ t)$$

is a p.d. kernel on  $S$ .

## Example: translation invariant kernels

$(\mathbb{R}^d, +, -)$  is an abelian group with involution. A function  $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}$  is p.d. if the function

$$K(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x} - \mathbf{y})$$

is p.d. on  $\mathbb{R}^d$  (translation invariant kernels).

# Semicharacters

## Definition

A function  $\rho : S \rightarrow \mathbb{C}$  on an **abelian** semigroup with involution  $(S, +, *)$  is called a **semicharacter** if

- 1  $\rho(0) = 1$ ,
- 2  $\rho(s + t) = \rho(s)\rho(t)$  for  $s, t \in S$ ,
- 3  $\rho(s^*) = \overline{\rho(s)}$  for  $s \in S$ .

The set of semicharacters on  $S$  is denoted by  $S^*$ .

## Remarks

- If  $*$  is the **identity**, a semicharacter is automatically **real-valued**.
- If  $(S, +)$  is an **abelian group** and  $s^* = -s$ , a semicharacter has its values in the circle group  $\{z \in \mathbb{C} \mid |z| = 1\}$  and is a **group character**.

# Semicharacters are p.d.

## Lemma

Every semicharacter is p.d., in the sense that:

- $K(s, t) = \overline{K(t, s)}$ ,
- $\sum_{i,j=1}^n a_i \bar{a}_j K(x_i, x_j) \geq 0$ .

## Proof

Direct from definition, e.g.,

$$\sum_{i,j=1}^n a_i \bar{a}_j \rho(x_i + x_j^*) = \sum_{i,j=1}^n a_i \bar{a}_j \rho(x_i) \overline{\rho(x_j)} \geq 0.$$

## Examples

- $\varphi(t) = e^{\beta t}$  on  $(\mathbb{R}, +, Id)$ .
- $\varphi(t) = e^{i\omega t}$  on  $(\mathbb{R}, +, -)$ .

# Integral representation of p.d. functions

## Theorem

Let  $(S, +, *)$  an abelian semigroup with involution. A function  $\varphi : S \rightarrow \mathbb{R}$  is p.d. and exponentially bounded (resp. bounded) if and only if it has a representation of the form:

$$\varphi(s) = \int_{S^*} \rho(s) d\mu(\rho).$$

where  $\mu$  is a Radon measure with compact support on  $S^*$  (resp. on  $\hat{S}$ , the set of bounded semicharacters).

# Integral representation of p.d. functions

## Definition

- An function  $\alpha : S \rightarrow \mathbb{R}$  on a semigroup with involution is called an **absolute value** if (i)  $\alpha(e) = 1$ , (ii)  $\alpha(s \circ t) \leq \alpha(s)\alpha(t)$ , and (iii)  $\alpha(s^*) = \alpha(s)$ .
- A function  $f : S \rightarrow \mathbb{R}$  is called **exponentially bounded** if there exists an absolute value  $\alpha$  and a constant  $C > 0$  s.t.  $|f(s)| \leq C\alpha(s)$  for  $s \in S$ .

## Theorem

Let  $(S, +, *)$  an abelian semigroup with involution. A function  $\varphi : S \rightarrow \mathbb{R}$  is p.d. and exponentially bounded (resp. bounded) if and only if it has a representation of the form:

$$\varphi(s) = \int_{S^*} \rho(s) d\mu(\rho).$$

where  $\mu$  is a Radon measure with compact support on  $S^*$  (resp. on  $\hat{S}$ , the set of bounded semicharacters).

# Proof

## Sketch (details in Berg et al., 1983, Theorem 4.2.5)

- For an absolute value  $\alpha$ , the set  $P_1^\alpha$  of  $\alpha$ -bounded p.d. functions that satisfy  $\varphi(0) = 1$  is a **compact convex set** whose **extreme points are precisely the  $\alpha$ -bounded semicharacters**.
- If  $\varphi$  is p.d. and exponentially bounded then there exists an absolute value  $\alpha$  such that  $\varphi(0)^{-1}\varphi \in P_1^\alpha$ .
- By the **Krein-Milman theorem** there exists a Radon probability measure on  $P_1^\alpha$  having  $\varphi(0)^{-1}\varphi$  as **barycentre**.

## Remarks

- The result is **not true** without the assumption of **exponentially bounded** semicharacters.
- In the case of abelian groups with  $s^* = -s$  this reduces to **Bochner's theorem** for discrete abelian groups, cf. Rudin (1962).

## Example 1: $(\mathbb{R}_+, +, Id)$

### Semicharacters

- $S = (\mathbb{R}_+, +, Id)$  is an **abelian semigroup**.
- P.d. functions are **nonnegative**, because  $\varphi(x) = \varphi(\sqrt{x})^2$ .
- The set of **bounded semicharacters** is exactly the set of functions:

$$s \in \mathbb{R}_+ \mapsto \rho_a(s) = e^{-as},$$

for  $a \in [0, +\infty]$  (left as exercise).

- **Non-bounded** semicharacters are more difficult to characterize; in fact there exist nonmeasurable solutions of the equation  $h(x + y) = h(x)h(y)$ .

## Example 1: $(\mathbb{R}_+, +, Id)$ (cont.)

### P.d. functions

- By the integral representation theorem for bounded semi-characters we obtain that a function  $\varphi : \mathbb{R}_+ \rightarrow \mathbb{R}$  is p.d. and bounded if and only if it has the form:

$$\varphi(s) = \int_0^\infty e^{-as} d\mu(a) + b\rho_\infty(s)$$

where  $\mu \in \mathcal{M}_+^b(\mathbb{R}_+)$  and  $b \geq 0$ .

- The first term is the Laplace transform of  $\mu$ .  $\varphi$  is p.d., bounded and continuous iff it is the Laplace transform of a measure in  $\mathcal{M}_+^b(\mathbb{R})$ .

## Example 2: Semigroup kernels for finite measures (1/6)

### Setting

- We assume that data to be processed are “bags-of-points”, i.e., sets of points (with repeats) of a space  $\mathcal{U}$ .
- **Example** : a finite-length string as a set of  $k$ -mers.
- How to define a **p.d. kernel between any two bags** that only depends on the **union of the bags**?
- See details and proofs in Cuturi et al. (2005).

## Example 2: Semigroup kernels for finite measures (2/6)

### Semigroup of bounded measures

- We can represent any bag-of-point  $\mathbf{x}$  as a finite measure on  $\mathcal{U}$ :

$$\mathbf{x} = \sum_i a_i \delta_{x_i},$$

where  $a_i$  is the number of occurrences on  $x_i$  in the bag.

- The measure that represents the union of two bags is the **sum of the measures** that represent each individual bag.
- This suggests to look at the semigroup  $(\mathcal{M}_+^b(\mathcal{U}), +, Id)$  of bounded Radon measures on  $\mathcal{U}$  and to search for p.d. functions  $\varphi$  on this semigroup.

## Example 2: Semigroup kernels for finite measures (3/6)

### Semicharacters

- For any Borel measurable function  $f : \mathcal{U} \rightarrow \mathbb{R}$  the function  $\rho_f : \mathcal{M}_+^b(\mathcal{U}) \rightarrow \mathbb{R}$  defined by:

$$\rho_f(\mu) = e^{\mu[f]}$$

is a semicharacter on  $(\mathcal{M}_+^b(\mathcal{U}), +)$ .

- Conversely,  $\rho$  is **continuous** semicharacter (for the topology of weak convergence) if and only if there exists a continuous function  $f : \mathcal{U} \rightarrow \mathbb{R}$  such that  $\rho = \rho_f$ .
- No such characterization for non-continuous characters, even bounded.

## Example 2: Semigroup kernels for finite measures (4/6)

### Corollary

Let  $\mathcal{U}$  be a Hausdorff space. For any **Radon measure**  $\mu \in \mathcal{M}_+^c(C(\mathcal{U}))$  with compact support on the Hausdorff space of continuous real-valued functions on  $\mathcal{U}$  endowed with the topology of pointwise convergence, the following function  $K$  is a **continuous p.d. kernel** on  $\mathcal{M}_+^b(\mathcal{U})$  (endowed with the topology of weak convergence):

$$K(\mu, \nu) = \int_{C(\mathcal{X})} e^{\mu[f] + \nu[f]} d\mu(f).$$

### Remarks

The converse is not true: there exist continuous p.d. kernels that do not have this integral representation (it might include non-continuous semicharacters)

## Example 2: Semigroup kernels for finite measures (5/6)

### Example : entropy kernel

- Let  $\mathcal{X}$  be the set of probability densities (w.r.t. some reference measure) on  $\mathcal{U}$  with finite entropy:

$$h(\mathbf{x}) = - \int_{\mathcal{U}} \mathbf{x} \ln \mathbf{x}.$$

- Then the following **entropy kernel** is a p.d. kernel on  $\mathcal{X}$  for all  $\beta > 0$ :

$$K(\mathbf{x}, \mathbf{x}') = e^{-\beta h\left(\frac{\mathbf{x} + \mathbf{x}'}{2}\right)}.$$

- Remark: only valid for **densities** (e.g., for a kernel density estimator from a bag-of-parts)

## Example 2: Semigroup kernels for finite measures (6/6)

### Examples : inverse generalized variance kernel

- Let  $\mathcal{U} = \mathbb{R}^d$  and  $\mathcal{M}_+^V(\mathcal{U})$  be the set of finite measure  $\mu$  with second order moment and non-singular variance

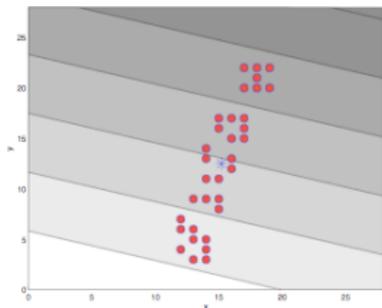
$$\Sigma(\mu) = \mu \left[ x x^\top \right] - \mu [x] \mu [x]^\top .$$

- Then the following function is a p.d. kernel on  $\mathcal{M}_+^V(\mathcal{U})$ , called the **inverse generalized variance kernel**:

$$K(\mu, \mu') = \frac{1}{\det \Sigma \left( \frac{\mu + \mu'}{2} \right)} .$$

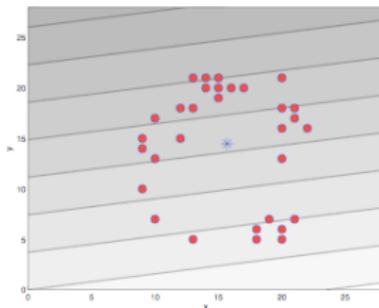
- Generalization possible with **regularization** and **kernel trick**.

# Application of semigroup kernel



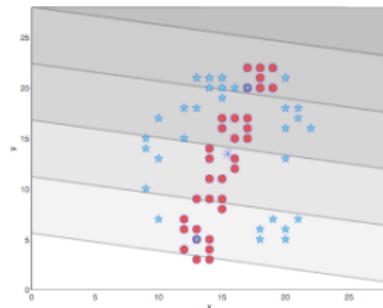
$$\Sigma_{1,1} = 0.0552$$

$$\Sigma_{2,2} = 0.0013$$



$$\Sigma'_{1,1} = 0.0441$$

$$\Sigma'_{2,2} = 0.0237$$



$$\Sigma''_{1,1} = 0.0497$$

$$\Sigma''_{2,2} = 0.0139$$

Weighted linear PCA of two different measures, with the first PC shown. Variances captured by the first and second PC are shown. The generalized variance kernel is the inverse of the product of the two values.

# Kernelization of the IGV kernel

## Motivations

- Gaussian distributions may be poor models.
- The method fails in large dimension

## Solution

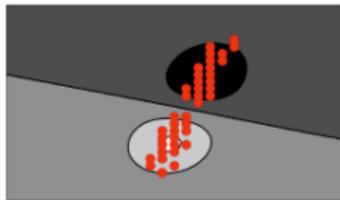
- 1 Regularization:

$$K_\lambda(\mu, \mu') = \frac{1}{\det\left(\Sigma\left(\frac{\mu+\mu'}{2}\right) + \lambda I_d\right)}.$$

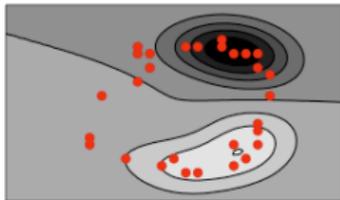
- 2 **Kernel trick**: the non-zero eigenvalues of  $UU^\top$  and  $U^\top U$  are the same  $\implies$  **replace the covariance matrix by the centered Gram matrix** (technical details in Cuturi et al., 2005).

# Illustration of kernel IGV kernel

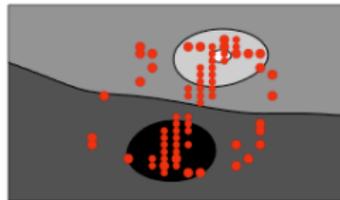
0.276



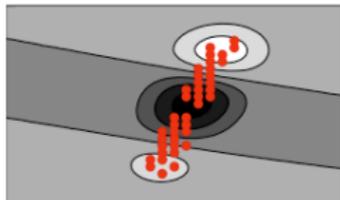
0.168



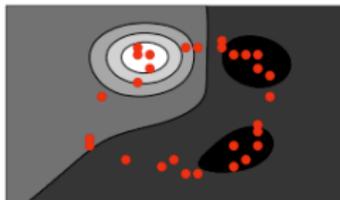
0.184



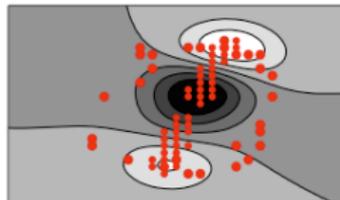
0.169



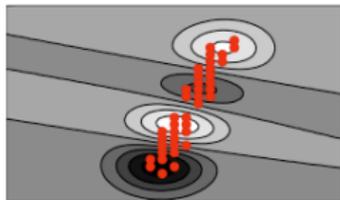
0.142



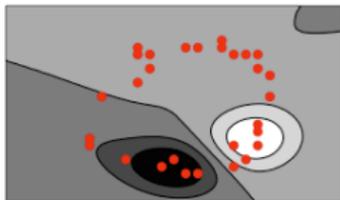
0.122



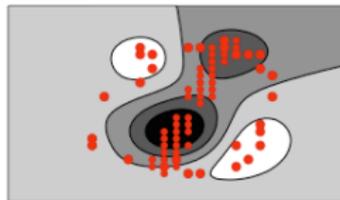
0.124



0.119



0.0934



# Semigroup kernel remarks

## Motivations

- A **very general formalism** to exploit an **algebraic structure** of the data.
- Kernel IVG kernel has given good results for character recognition from a subsampled image.
- The main motivation is more generally to develop kernels for **complex objects** from which **simple “patches” can be extracted**.
- The extension to **nonabelian groups** (e.g., permutation in the symmetric group) might find natural applications.

# Outline

## 4 The Kernel Jungle

- Mercer kernels and shift-invariant kernels
  - Shift-invariant kernels
  - Generalization to semigroups
  - Mercer kernels
    - RKHS and Green functions
- Kernels for generative models
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs

# Mercer kernels

## Definition

A kernel  $K$  on a set  $\mathcal{X}$  is called a **Mercer kernel** if:

- 1  $\mathcal{X}$  is a **compact metric space** (typically, a closed bounded subset of  $\mathbb{R}^d$ ).
- 2  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is a **continuous** p.d. kernel (w.r.t. the Borel topology)

## Motivations

- We can exhibit an explicit and intuitive feature space for a large class of p.d. kernels
- Historically, provided the first proof that a p.d. kernel is an inner product for non-finite sets  $\mathcal{X}$  (Mercer, 1905).
- Can be thought of as the natural generalization of the factorization of positive semidefinite matrices over infinite spaces.

## Sketch of the proof

- 1 The kernel matrix when  $\mathcal{X}$  is finite becomes a **linear operator** when  $\mathcal{X}$  is a metric space.
- 2 The matrix was positive semidefinite in the finite case, the linear operator is **self-adjoint** and **positive** in the metric case.
- 3 The **spectral theorem** states that any **compact** linear operator admits a complete orthonormal basis of eigenfunctions, with non-negative eigenvalues (just like positive semidefinite matrices can be diagonalized with nonnegative eigenvalues).
- 4 The kernel function can then be expanded over basis of eigenfunctions as:

$$K(\mathbf{x}, \mathbf{t}) = \sum_{k=1}^{\infty} \lambda_k \psi_k(\mathbf{x}) \psi_k(\mathbf{t}),$$

where  $\lambda_j \geq 0$  are the non-negative eigenvalues.

## In case of...

### Definition

Let  $\mathcal{H}$  be a Hilbert space

- A **linear operator** is a continuous linear mapping from  $\mathcal{H}$  to itself.
- A linear operator  $L$  is called **compact** if, for any bounded sequence  $\{f_n\}_{n=1}^{\infty}$ , the sequence  $\{Lf_n\}_{n=1}^{\infty}$  has a subsequence that converges.
- $L$  is called **self-adjoint** if, for any  $f, g \in \mathcal{H}$ :

$$\langle f, Lg \rangle = \langle Lf, g \rangle .$$

- $L$  is called **positive** if it is self-adjoint and, for any  $f \in \mathcal{H}$ :

$$\langle f, Lf \rangle \geq 0 .$$

# An important lemma

## The linear operator

- Let  $\nu$  be **any** Borel measure on  $\mathcal{X}$ , and  $L_2^\nu(\mathcal{X})$  the Hilbert space of square integrable functions on  $\mathcal{X}$ .
- For any function  $K : \mathcal{X}^2 \mapsto \mathbb{R}$ , let the transform:

$$\forall f \in L_2^\nu(\mathcal{X}), \quad (L_K f)(\mathbf{x}) = \int K(\mathbf{x}, \mathbf{t}) f(\mathbf{t}) d\nu(\mathbf{t}).$$

## Lemma

If  $K$  is a Mercer kernel, then  $L_K$  is a **compact and bounded linear operator** over  $L_2^\nu(\mathcal{X})$ , **self-adjoint and positif**.

## Proof (1/6)

$L_K$  is a mapping from  $L^{\nu}_2(\mathcal{X})$  to  $L^{\nu}_2(\mathcal{X})$

For any  $f \in L^{\nu}_2(\mathcal{X})$  and  $(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{X}^2$ :

$$\begin{aligned} |L_K f(\mathbf{x}_1) - L_K f(\mathbf{x}_2)| &= \left| \int (K(\mathbf{x}_1, \mathbf{t}) - K(\mathbf{x}_2, \mathbf{t})) f(\mathbf{t}) d\nu(\mathbf{t}) \right| \\ &\leq \|K(\mathbf{x}_1, \cdot) - K(\mathbf{x}_2, \cdot)\| \|f\| \\ &\quad \text{(Cauchy-Schwarz)} \\ &\leq \sqrt{\nu(\mathcal{X})} \max_{\mathbf{t} \in \mathcal{X}} |K(\mathbf{x}_1, \mathbf{t}) - K(\mathbf{x}_2, \mathbf{t})| \|f\|. \end{aligned}$$

$K$  being continuous and  $\mathcal{X}$  compact,  $K$  is uniformly continuous, therefore  $L_K f$  is continuous. In particular,  $L_K f \in L^{\nu}_2(\mathcal{X})$  (with the slight abuse of notation  $\mathcal{C}(\mathcal{X}) \subset L^{\nu}_2(\mathcal{X})$ ).  $\square$

## Proof (2/6)

### $L_K$ is linear and continuous

- Linearity is obvious (by definition of  $L_K$  and linearity of the integral).
- For continuity, we observe that for all  $f \in L^2_\nu(\mathcal{X})$  and  $\mathbf{x} \in \mathcal{X}$ :

$$\begin{aligned} |(L_K f)(\mathbf{x})| &= \left| \int K(\mathbf{x}, \mathbf{t}) f(\mathbf{t}) d\nu(\mathbf{t}) \right| \\ &\leq \sqrt{\nu(\mathcal{X})} \max_{\mathbf{t} \in \mathcal{X}} |K(\mathbf{x}, \mathbf{t})| \|f\| \\ &\leq \sqrt{\nu(\mathcal{X})} C_K \|f\|. \end{aligned}$$

with  $C_K = \max_{\mathbf{x}, \mathbf{t} \in \mathcal{X}} |K(\mathbf{x}, \mathbf{t})|$ . Therefore:

$$\|L_K f\| = \left( \int L_K f(\mathbf{t})^2 d\nu(\mathbf{t}) \right)^{\frac{1}{2}} \leq \nu(\mathcal{X}) C_K \|f\|. \quad \square$$

## Proof (3/6)

### Criterion for compactity

In order to prove the compactity of  $L_K$  we need the following criterion. Let  $C(\mathcal{X})$  denote the set of continuous functions on  $\mathcal{X}$  endowed with infinite norm  $\|f\|_\infty = \max_{\mathbf{x} \in \mathcal{X}} |f(\mathbf{x})|$ .

A set of functions  $G \subset C(\mathcal{X})$  is called **equicontinuous** if:

$$\forall \epsilon > 0, \exists \delta > 0, \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2, \\ \|\mathbf{x} - \mathbf{y}\| < \delta \implies \forall g \in G, |g(\mathbf{x}) - g(\mathbf{y})| < \epsilon.$$

### Ascoli Theorem

A part  $H \subset C(\mathcal{X})$  is **relatively compact** (i.e., its closure is compact) if and only if it is **uniformly bounded** and **equicontinuous**.

## Proof (4/6)

### $L_K$ is compact

Let  $(f_n)_{n \geq 0}$  be a bounded sequence of  $L_2^\nu(\mathcal{X})$  ( $\|f_n\| < M$  for all  $n$ ). The sequence  $(L_K f_n)_{n \geq 0}$  is a sequence of continuous functions, uniformly bounded because:

$$\|L_K f\|_\infty \leq \sqrt{\nu(\mathcal{X})} C_K \|f\| \leq \sqrt{\nu(\mathcal{X})} C_K M.$$

It is equicontinuous because:

$$|L_K f_n(\mathbf{x}_1) - L_K f_n(\mathbf{x}_2)| \leq \sqrt{\nu(\mathcal{X})} \max_{\mathbf{t} \in \mathcal{X}} |K(\mathbf{x}_1, \mathbf{t}) - K(\mathbf{x}_2, \mathbf{t})| M.$$

By Ascoli theorem, we can extract a sequence uniformly convergent in  $C(\mathcal{X})$ , and therefore in  $L_2^\nu(\mathcal{X})$ .  $\square$

## Proof (5/6)

$L_K$  is self-adjoint

$K$  being symmetric, we have for all  $f, g \in \mathcal{H}$ :

$$\begin{aligned}\langle f, Lg \rangle &= \int f(\mathbf{x})(Lg)(\mathbf{x})\nu(d\mathbf{x}) \\ &= \int \int f(\mathbf{x})g(\mathbf{t})K(\mathbf{x},\mathbf{t})\nu(d\mathbf{x})\nu(d\mathbf{t}) \quad (\text{Fubini}) \\ &= \langle Lf, g \rangle.\end{aligned}$$

## Proof (6/6)

$L_K$  is positif

We can approximate the integral by finite sums:

$$\begin{aligned}\langle f, Lf \rangle &= \int \int f(\mathbf{x}) f(\mathbf{t}) K(\mathbf{x}, \mathbf{t}) \nu(d\mathbf{x}) \nu(d\mathbf{t}) \\ &= \lim_{k \rightarrow \infty} \frac{\nu(\mathcal{X})}{k^2} \sum_{i,j=1}^k K(\mathbf{x}_i, \mathbf{x}_j) f(\mathbf{x}_i) f(\mathbf{x}_j) \\ &\geq 0,\end{aligned}$$

because  $K$  is positive definite.  $\square$

# Diagonalization of the operator

We need the following general result:

## Spectral theorem

Let  $L$  be a **compact** linear operator on a Hilbert space  $\mathcal{H}$ . Then there exists in  $\mathcal{H}$  a **complete orthonormal system**  $(\psi_1, \psi_2, \dots)$  of eigenvectors of  $L$ . The eigenvalues  $(\lambda_1, \lambda_2, \dots)$  are **real** if  $L$  is self-adjoint, and **non-negative** if  $L$  is positive.

## Remark

This theorem can be applied to  $L_K$ . In that case the eigenfunctions  $\varphi_k$  associated to the eigenvalues  $\lambda_k \neq 0$  can be considered as **continuous functions**, because:

$$\psi_k = \frac{1}{\lambda_k} L \psi_K.$$

# Main result

## Mercer Theorem

Let  $\mathcal{X}$  be a compact metric space,  $\nu$  a Borel measure on  $\mathcal{X}$ , and  $K$  a continuous p.d. kernel. Let  $(\lambda_1, \lambda_2, \dots)$  denote the nonnegative eigenvalues of  $L_K$  and  $(\psi_1, \psi_2, \dots)$  the corresponding eigenfunctions. Then all  $\psi_k$  are continuous functions, and for any  $\mathbf{x}, \mathbf{t} \in \mathcal{X}$ :

$$K(\mathbf{x}, \mathbf{t}) = \sum_{k=1}^{\infty} \lambda_k \psi_k(\mathbf{x}) \psi_k(\mathbf{t}),$$

where the convergence is absolute for each  $\mathbf{x}, \mathbf{t} \in \mathcal{X}$ , and uniform on  $\mathcal{X} \times \mathcal{X}$ .

# Mercer kernels as inner products

## Corollary

The mapping

$$\begin{aligned}\Phi : \mathcal{X} &\mapsto l^2 \\ \mathbf{x} &\mapsto \left( \sqrt{\lambda_k} \psi_k(\mathbf{x}) \right)_{k \in \mathbb{N}}\end{aligned}$$

is well defined, continuous, and satisfies

$$K(\mathbf{x}, \mathbf{t}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{t}) \rangle_{l^2} .$$

# Proof of the corollary

## Proof

By Mercer theorem we see that for all  $\mathbf{x} \in \mathcal{X}$ ,  $\sum \lambda_k \psi_k^2(\mathbf{x})$  converges to  $K(\mathbf{x}, \mathbf{x}) < \infty$ , therefore  $\Phi(\mathbf{x}) \in l^2$ .

The continuity of  $\Phi$  results from:

$$\begin{aligned} \|\Phi(\mathbf{x}) - \Phi(\mathbf{t})\|_{l^2}^2 &= \sum_{k=1}^{\infty} \lambda_k (\psi_k(\mathbf{x}) - \psi_k(\mathbf{t}))^2 \\ &= K(\mathbf{x}, \mathbf{x}) + K(\mathbf{t}, \mathbf{t}) - 2K(\mathbf{x}, \mathbf{t}) \end{aligned}$$

# Summary

- This proof extends the proof valid when  $\mathcal{X}$  is finite.
- This is a **constructive** proof, developed by Mercer (1905).
- **Compactness** and **continuity** are required. For instance, for  $\mathcal{X} = \mathbb{R}^d$ , the eigenvalues of:

$$\int_{\mathcal{X}} K(\mathbf{x}, \mathbf{t}) \psi(\mathbf{t}) = \lambda \psi(\mathbf{t})$$

are not necessarily countable, Mercer theorem does not hold. Other tools are thus required such as the Fourier transform for shift-invariant kernels.

## RKHS of Mercer kernels

- Let  $\mathcal{X}$  be a compact metric space, and  $K$  a **Mercer kernel** on  $\mathcal{X}$  (symmetric, continuous and positive definite).
- We have expressed a decomposition of the kernel in terms of the **eigenfunctions** of the linear **convolution operator**.
- In some cases this provides an **intuitive** feature space.
- The kernel also has a **RKHS**, like any p.d. kernel.
- Can we get an **intuition of the RKHS norm** in terms of the **eigenfunctions and eigenvalues** of the convolution operator?

## Reminder: expansion of Mercer kernel

### Theorem

Denote by  $L_K$  the linear operator of  $L_2^\nu(\mathcal{X})$  defined by:

$$\forall f \in L_2^\nu(\mathcal{X}), (L_K f)(\mathbf{x}) = \int K(\mathbf{x}, \mathbf{t}) f(\mathbf{t}) d\nu(\mathbf{t}).$$

Let  $(\lambda_1, \lambda_2, \dots)$  denote the eigenvalues of  $L_K$  in decreasing order, and  $(\psi_1, \psi_2, \dots)$  the corresponding eigenfunctions. Then it holds that for any  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ :

$$K(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{\infty} \lambda_k \psi_k(\mathbf{x}) \psi_k(\mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{l^2},$$

with  $\Phi : \mathcal{X} \mapsto l^2$  defined par  $\Phi(\mathbf{x}) = (\sqrt{\lambda_k} \psi_k(\mathbf{x}))_{k \in \mathbb{N}}$ .

# RKHS construction

## Theorem

Assuming that all eigenvalues are positive, the RKHS is the Hilbert space:

$$H_K = \left\{ f \in L_2^\nu(\mathcal{X}) : f = \sum_{i=1}^{\infty} a_i \psi_i, \quad \text{with } \sum_{k=1}^{\infty} \frac{a_k^2}{\lambda_k} < \infty \right\}$$

endowed with the inner product:

$$\langle f, g \rangle_K = \sum_{k=1}^{\infty} \frac{a_k b_k}{\lambda_k}, \quad \text{for } f = \sum_k a_k \psi_k, g = \sum_k b_k \psi_k.$$

## Remark

If some eigenvalues are equal to zero, then the result and the proof remain valid on the subspace spanned by the eigenfunctions with positive eigenvalues.

# Proof (1/6)

## Sketch

In order to show that  $H_K$  is the RKHS of the kernel  $K$  we need to show that:

- 1 it is a **Hilbert space of functions** from  $\mathcal{X}$  to  $\mathbb{R}$ ,
- 2 for any  $\mathbf{x} \in \mathcal{X}$ ,  $K_{\mathbf{x}} \in H_K$ ,
- 3 for any  $\mathbf{x} \in \mathcal{X}$  and  $f \in H_K$ ,  $f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{H_K}$ .

## Proof (2/6)

$H_K$  is a Hilbert space

Indeed the function:

$$L_K^{\frac{1}{2}} : L_2^{\nu}(\mathcal{X}) \rightarrow H_K$$
$$\sum_{i=1}^{\infty} a_i \psi_i \mapsto \sum_{i=1}^{\infty} a_i \sqrt{\lambda_i} \psi_i$$

is an isomorphism, therefore  $H_K$  is a Hilbert space, like  $L_2^{\nu}(\mathcal{X})$ .  $\square$

## Proof (3/6)

$H_K$  is a space of continuous functions

For any  $f = \sum_{i=1}^{\infty} a_i \psi_i \in H_K$ , and  $\mathbf{x} \in \mathcal{X}$ , we have (if  $f(x)$  makes sense):

$$\begin{aligned} |f(\mathbf{x})| &= \left| \sum_{i=1}^{\infty} a_i \psi_i(\mathbf{x}) \right| = \left| \sum_{i=1}^{\infty} \frac{a_i}{\sqrt{\lambda_i}} \sqrt{\lambda_i} \psi_i(\mathbf{x}) \right| \\ &\leq \left( \sum_{i=1}^{\infty} \frac{a_i^2}{\lambda_i} \right)^{\frac{1}{2}} \cdot \left( \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x})^2 \right)^{\frac{1}{2}} \\ &= \|f\|_{H_K} K(\mathbf{x}, \mathbf{x})^{\frac{1}{2}} \\ &= \|f\|_{H_K} \sqrt{C_K}. \end{aligned}$$

Therefore **convergence in  $\|\cdot\|_{H_K}$  implies uniform convergence** for functions.

## Proof (4/6)

$H_K$  is a space of continuous functions (cont.)

Let now  $f_n = \sum_{i=1}^n a_i \psi_i \in H_K$ . The functions  $\psi_i$  are continuous functions, therefore  $f_n$  is also continuous, for all  $n$ . The  $f_n$ 's are convergent in  $H_K$ , therefore also in the (complete) space of continuous functions endowed with the uniform norm.

Let  $f_c$  the continuous limit function. Then  $f_c \in L_2^\nu(\mathcal{X})$  and

$$\|f_n - f_c\|_{L_2^\nu(\mathcal{X})} \xrightarrow{n \rightarrow \infty} 0.$$

On the other hand,

$$\|f - f_n\|_{L_2^\nu(\mathcal{X})} \leq \lambda_1 \|f - f_n\|_{H_K} \xrightarrow{n \rightarrow \infty} 0,$$

therefore  $f = f_c$ .  $\square$

## Proof (5/6)

$$K_x \in H_K$$

For any  $\mathbf{x} \in \mathcal{X}$  let, for all  $i$ ,  $a_i = \lambda_i \psi_i(\mathbf{x})$ . We have:

$$\sum_{i=1}^{\infty} \frac{a_i^2}{\lambda_i} = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x})^2 = K(\mathbf{x}, \mathbf{x}) < \infty,$$

therefore  $\varphi_x := \sum_{i=1}^{\infty} a_i \psi_i \in H_K$ . As seen earlier the convergence in  $H_K$  implies pointwise convergence, therefore for any  $\mathbf{t} \in \mathcal{X}$ :

$$\varphi_x(\mathbf{t}) = \sum_{i=1}^{\infty} a_i \psi_i(\mathbf{t}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{t}) = K(\mathbf{x}, \mathbf{t}),$$

therefore  $\varphi_x = K_x \in H_K$ .  $\square$

## Proof (6/6)

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{H_K}$$

Let  $f = \sum_{i=1}^{\infty} a_i \psi_i \in H_K$ , et  $\mathbf{x} \in \mathcal{X}$ . We have seen that:

$$K_{\mathbf{x}} = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i,$$

therefore:

$$\langle f, K_{\mathbf{x}} \rangle_{H_K} = \sum_{i=1}^{\infty} \frac{\lambda_i \psi_i(\mathbf{x}) a_i}{\lambda_i} = \sum_{i=1}^{\infty} a_i \psi_i(\mathbf{x}) = f(\mathbf{x}),$$

which concludes the proof.  $\square$

## Remarks

- Although  $H_K$  was built from the eigenfunctions of  $L_K$ , which depend on the choice of the measure  $\nu(\mathbf{x})$ , we know by unicity of the RKHS that  $H_K$  is independant of  $\nu$  and  $L_K$ .
- Mercer theorem provides a **concrete way** to build the RKHS, by taking linear combinations of the eigenfunctions of  $L_K$  (with adequately chosen weights).
- The eigenfunctions  $(\psi_i)_{i \in \mathbb{N}}$  form an **orthogonal basis** of the RKHS:

$$\langle \psi_i, \psi_j \rangle_{H_K} = 0 \quad \text{si } i \neq j, \quad \|\psi_i\|_{H_K} = \frac{1}{\sqrt{\lambda_i}}.$$

The RKHS is a well-defined **ellipsoid** with axes given by the eigenfunctions.

# Outline

## 4 The Kernel Jungle

- Mercer kernels and shift-invariant kernels
  - Shift-invariant kernels
  - Generalization to semigroups
  - Mercer kernels
  - RKHS and Green functions
- Kernels for generative models
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs

# Motivations

- The RKHS norm is related to the **smoothness** of functions.
- Smoothness of a function is naturally quantified by **Sobolev norms** (in particular  $L_2$  norms of derivatives).
- In this section we make a general link between RKHS and **Green functions defined by differential operators**.

# A simple example

## Explicit choice of smoothness

Let

$$\mathcal{H} = \{f : [0, 1] \mapsto \mathbb{R}, \text{absolutely continuous, } f' \in L^2([0, 1]), f(0) = 0\} .$$

endowed with the bilinear form:

$$\forall (f, g) \in \mathcal{F}^2 \quad \langle f, g \rangle_{\mathcal{H}} = \int_0^1 f'(u) g'(u) du .$$

Note that  $\langle f, f \rangle_{\mathcal{H}}$  measures the smoothness of  $f$ :

$$\langle f, f \rangle_{\mathcal{H}} = \int_0^1 f'(u)^2 du = \|f'\|_{L^2([0,1])}^2 .$$

# The RKHS point of view

## Theorem

$\mathcal{H}$  is a RKHS with r.k. given by:

$$\forall (x, y) \in [0, 1]^2, \quad K(x, y) = \min(x, y).$$

## Remark

Therefore,  $\|f\|_{\mathcal{H}} = \|f'\|_{L^2}$ : the RKHS norm is precisely the smoothness functional defined in the simple example.

# Proof (1/3)

## Sketch

We need to show that

- $\mathcal{H}$  is a Hilbert space
- $\forall x \in [0, 1], K_x \in \mathcal{H},$
- $\forall (x, f) \in [0, 1] \times \mathcal{H}, \langle f, K_x \rangle_{\mathcal{H}} = f(x).$

# Proof (1/3)

## Sketch

We need to show that

- $\mathcal{H}$  is a Hilbert space
- $\forall x \in [0, 1], K_x \in \mathcal{H},$
- $\forall (x, f) \in [0, 1] \times \mathcal{H}, \langle f, K_x \rangle_{\mathcal{H}} = f(x).$

## Proof (2/3)

### $\mathcal{H}$ is a pre-Hilbert space

- $f$  absolutely continuous implies differentiable almost everywhere, and

$$\forall x \in [0, 1], \quad f(x) = f(0) + \int_0^x f'(u) du .$$

- For any  $f \in \mathcal{H}$ ,  $f(0) = 0$  implies by Cauchy-Schwarz:

$$|f(x)| = \left| \int_0^x f'(u) du \right| \leq \sqrt{x} \left( \int_0^1 f'(u)^2 du \right)^{\frac{1}{2}} = \sqrt{x} \|f\|_{\mathcal{H}} .$$

Therefore,  $\|f\|_{\mathcal{H}} = 0 \implies f = 0$ , showing that  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  is an inner product.  $\mathcal{H}$  is thus a pre-Hilbert space.

## Proof (2/3)

### $\mathcal{H}$ is a Hilbert space

- To show that  $\mathcal{H}$  is complete, let  $(f_n)_{n \in \mathbb{N}}$  a Cauchy sequence in  $\mathcal{H}$
- $(f'_n)_{n \in \mathbb{N}}$  is a Cauchy sequence in  $L^2[0, 1]$ , thus converges to  $g \in L^2[0, 1]$
- By the previous inequality,  $(f_n(x))_{n \in \mathbb{N}}$  is a Cauchy sequence and thus converges to a real number  $f(x)$ , for any  $x \in [0, 1]$ . Moreover:

$$f(x) = \lim_n f_n(x) = \lim_n \int_0^x f'_n(u) du = \int_0^x g(u) du,$$

showing that  $f$  is absolutely continuous and  $f' = g$  almost everywhere; in particular,  $f' \in L^2[0, 1]$ .

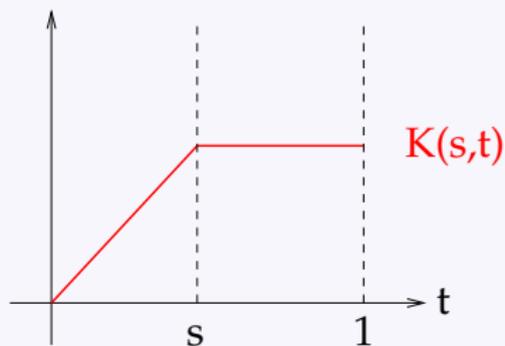
- Finally,  $f(0) = \lim_n f_n(0) = 0$ , therefore  $f \in \mathcal{H}$  and

$$\lim_n \|f_n - f\|_{\mathcal{H}} = \|f' - g_n\|_{L^2[0,1]} = 0.$$

## Proof (2/3)

$\forall x \in [0, 1], K_x \in \mathcal{H}$

Let  $K_x(y) = K(x, y) = \min(x, y)$  sur  $[0, 1]^2$ :



$K_x$  is differentiable except at  $s$ , has a square integrable derivative, and  $K_x(0) = 0$ , therefore  $K_x \in \mathcal{H}$  for all  $x \in [0, 1]$ .  $\square$

## Proof (3/3)

For all  $x, f$ ,  $\langle f, K_x \rangle_{\mathcal{H}} = f(x)$

For any  $x \in [0, 1]$  and  $f \in \mathcal{H}$  we have:

$$\langle f, K_x \rangle_{\mathcal{H}} = \int_0^1 f'(u) K'_x(u) du = \int_0^x f'(u) du = f(x),$$

which shows that  $K$  is the r.k. associated to  $\mathcal{H}$ .  $\square$

# Generalization

## Theorem

Let  $\mathcal{X} = \mathbb{R}^d$  and  $D$  a differential operator on a class of functions  $\mathcal{H}$  such that, endowed with the inner product:

$$\forall (f, g) \in \mathcal{H}^2, \quad \langle f, g \rangle_{\mathcal{H}} = \langle Df, Dg \rangle_{L^2(\mathcal{X})},$$

it is a Hilbert space.

Then  $\mathcal{H}$  is a RKHS that admits as r.k. the Green function of the operator  $D^*D$ , where  $D^*$  denotes the adjoint operator of  $D$ .

## In case of...

### Green functions

Let the differential equation on  $\mathcal{H}$ :

$$f = Dg,$$

where  $g$  is unknown. In order to solve it we can look for  $g$  of the form:

$$g(x) = \int_{\mathcal{X}} k(x, y) f(y) dy$$

for some function  $k : \mathcal{X}^2 \mapsto \mathbb{R}$ .  $k$  must then satisfy, for all  $x \in \mathcal{X}$ ,

$$f(x) = Dg(x) = \langle Dk_x, f \rangle_{L^2(\mathcal{X})}.$$

$k$  is called the **Green function** of the operator  $D$ .

## Proof

Let  $\mathcal{H}$  be a Hilbert space endowed with the inner product:

$$\langle f, g \rangle_{\mathcal{H}} = \langle Df, Dg \rangle_{L^2(\mathcal{X})},$$

and  $K$  be the Green function of the operator  $D^*D$ . For all  $x \in \mathcal{X}$ ,  $K_x \in \mathcal{H}$  because:

$$\langle DK_x, DK_x \rangle_{L^2(\mathcal{X})} = \langle D^*DK_x, K_x \rangle_{L^2(\mathcal{X})} = K_x(x) < \infty.$$

Moreover, for all  $f \in \mathcal{H}$  and  $x \in \mathcal{X}$ , we have:

$$f(x) = \langle D^*DK_x, f \rangle_{L^2(\mathcal{X})} = \langle DK_x, Df \rangle_{L^2(\mathcal{X})} = \langle K_x, f \rangle_{\mathcal{H}},$$

which shows that  $\mathcal{H}$  is a RKHS with  $K$  as r.k.  $\square$

## Kernel examples: Summary

- Many notions of **smoothness** can be translated as **RKHS norms** for particular kernels (eigenvalues convolution operator, Sobolev norms and Green operators, Fourier transforms...).
- There is **no “uniformly best kernel”**, but rather a large **toolbox** of methods and tricks to **encode prior knowledge** and exploit the **nature or structure** of the data.
- In the following sections we focus on **particular data and applications** to illustrate the process of **kernel design**.

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - **Kernels for generative models**
  - Kernels for biological sequences
  - Kernels for graphs
  - Kernels on graphs

# Motivation

Kernel methods are sometimes criticized for their **lack of flexibility**: a large effort is spent in designing by hand the kernel.

## Question

How do we design a kernel **adapted** to the data?

# Motivation

Kernel methods are sometimes criticized for their **lack of flexibility**: a large effort is spent in designing by hand the kernel.

## Question

How do we design a kernel **adapted** to the data?

## Answer

A successful strategy is given by kernels for generative models, which are/have been the state of the art in many fields, including image and sequence representations.

## Parametric model

A **model** is a family of distribution

$$\{P_\theta, \theta \in \Theta \subset \mathbb{R}^m\} \subseteq \mathcal{M}_1^+(\mathcal{X}).$$

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
    - Fisher kernel
      - Mutual information kernels
      - Marginalized kernels
  - Kernels for biological sequences
  - Kernels for graphs
  - Kernels on graphs

# Fisher kernel

## Definition

- Fix a parameter  $\theta_0 \in \Theta$  (e.g., by maximum likelihood over a training set of sequences)
- For each sequence  $\mathbf{x}$ , compute the **Fisher score vector**:

$$\Phi_{\theta_0}(\mathbf{x}) = \nabla_{\theta} \log P_{\theta}(\mathbf{x})|_{\theta=\theta_0} .$$

- Form the kernel (Jaakkola et al., 2000):

$$K(\mathbf{x}, \mathbf{x}') = \Phi_{\theta_0}(\mathbf{x})^{\top} \mathbf{I}(\theta_0)^{-1} \Phi_{\theta_0}(\mathbf{x}') ,$$

where  $\mathbf{I}(\theta_0) = E [\Phi_{\theta_0}(\mathbf{x})\Phi_{\theta_0}(\mathbf{x})^{\top}]$  is the Fisher information matrix.

## Fisher kernel properties (1/2)

- The Fisher score describes how **each parameter contributes** to the process of generating a particular example
- A kernel classifier employing the Fisher kernel derived from a model that contains the label as a latent variable is, asymptotically, **at least as good a classifier as the MAP labelling** based on the model (Jaakkola and Haussler, 1999).
- A variant of the Fisher kernel (called the Tangent of Posterior kernel) can also improve over the direct posterior classification by helping to **correct the effect of estimation errors** in the parameter (Tsuda et al., 2002).

## Fisher kernel properties (2/2)

### Lemma

The Fisher kernel is **invariant** under change of parametrization.

- Consider indeed different parametrization given by some diffeomorphism  $\lambda = f(\theta)$ . The **Jacobian** matrix relating the parametrization is denoted by  $[\mathbf{J}]_{ij} = \frac{\partial \theta_j}{\partial \lambda_i}$ .
- The gradient of log-likelihood w.r.t. to the new parameters is

$$\Phi_{\lambda_0}(\mathbf{x}) = \nabla_{\lambda} \log P_{\lambda_0}(\mathbf{x}) = \mathbf{J} \nabla_{\theta} \log P_{\theta_0}(\mathbf{x}) = \mathbf{J} \Phi_{\theta_0}(\mathbf{x}).$$

- the Fisher information matrix is

$$\mathbf{I}(\lambda_0) = E \left[ \Phi_{\theta_0}(\mathbf{x}) \Phi_{\theta_0}(\mathbf{x})^{\top} \right] = \mathbf{J} \mathbf{I}(\theta_0) \mathbf{J}^{\top}.$$

- we conclude by noticing that  $\mathbf{I}(\lambda_0)^{-1} = \mathbf{J}^{-1} \mathbf{I}(\theta_0)^{-1} \mathbf{J}^{\top -1}$ .

## Fisher kernel in practice

- $\Phi_{\theta_0}(\mathbf{x})$  can be computed explicitly for many models (e.g., HMMs), where the model is **first estimated from data**.
- $\mathbf{I}(\theta_0)$  is often replaced by the identity matrix for simplicity.
- Several different models (i.e., different  $\theta_0$ ) can be trained and combined.
- The **Fisher vectors** are defined as  $\varphi_{\theta_0}(\mathbf{x}) = \mathbf{I}(\theta_0)^{-1/2} \Phi_{\theta_0}(\mathbf{x})$ . They are explicitly computed and correspond to an explicit embedding:  
 $K(\mathbf{x}, \mathbf{x}') = \varphi_{\theta_0}(\mathbf{x})^\top \varphi_{\theta_0}(\mathbf{x}')$ .

## Fisher kernels: example with Gaussian data model (1/2)

Consider a normal distribution  $\mathcal{N}(\mu, \sigma^2)$  and denote by  $\alpha = 1/\sigma^2$  the inverse variance, i.e., precision parameter. With  $\theta = (\mu, \alpha)$ , we have

$$\log P_{\theta}(x) = \frac{1}{2} \log \alpha - \frac{1}{2} \log(2\pi) - \frac{1}{2} \alpha (x - \mu)^2,$$

and thus

$$\frac{\partial \log P_{\theta}(x)}{\partial \mu} = \alpha(x - \mu), \quad \frac{\partial \log P_{\theta}(x)}{\partial \alpha} = \frac{(x - \mu)^2}{2\alpha},$$

and

$$\mathbf{I}(\theta) = \begin{pmatrix} \alpha & 0 \\ 0 & (1/2)\alpha^{-2} \end{pmatrix}.$$

The **Fisher vector** is then

$$\varphi_{\theta}(\mathbf{x}) = \begin{pmatrix} (x - \mu)/\sigma \\ (1/\sqrt{2})(1 - (x - \mu)^2/\sigma^2) \end{pmatrix}.$$

## Fisher kernels: example with Gaussian data model (2/2)

Now consider an i.i.d. data model over a set of data points  $x_1, \dots, x_n$  all distributed according to  $\mathcal{N}(\mu, \sigma^2)$ :

$$P_\theta(x_1, \dots, x_n) = \prod_{i=1}^n P_\theta(x_i).$$

Then, the Fisher vector is given by the sum of Fisher vectors of the points.

- Encodes the **discrepancy in the first and second order moment** of the data w.r.t. those of the model.

$$\varphi(x_1, \dots, x_n) = \sum_{i=1}^n \varphi(x_i) = n \begin{pmatrix} (\hat{\mu} - \mu)/\sigma \\ (\sigma^2 - \hat{\sigma}^2)/(\sqrt{2}\sigma^2) \end{pmatrix},$$

- where

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i \quad \text{and} \quad \hat{\sigma} = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{\mu})^2.$$

## Application: Aggregation of visual words (1/4)

- **Patch extraction and description stage:**

In various contexts, images may be described as a set of patches  $\mathbf{x}_1, \dots, \mathbf{x}_n$  computed at interest points. For example, SIFT, HOG, LBP, color histograms, convolutional features...

- **Coding stage:** The set of patches is then encoded into a single representation  $\varphi(\mathbf{x}_i)$ , typically in a high-dimensional space.
- **Pooling stage:** For example, sum pooling

$$\varphi(\mathbf{x}_1, \dots, \mathbf{x}_n) = \sum_{i=1}^n \varphi(\mathbf{x}_i).$$

Fisher vectors with a Gaussian Mixture Model (GMM) is considered to be a state-of-the-art aggregation technique [Perronnin and Dance, 2007].

## Application: Aggregation of visual words (2/4)

Let  $\theta = (\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)_{j=1 \dots k}$  be the parameters of a GMM with  $k$  Gaussian components. Then, the probabilistic model is given by

$$P_{\theta}(\mathbf{x}) = \sum_{j=1}^k \pi_j \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j).$$

### Remarks

- Each mixture component corresponds to a **visual word**, with a mean, variance, and mixing weight.
- Diagonal covariances  $\boldsymbol{\Sigma}_j = \text{diag}(\sigma_{j1}, \dots, \sigma_{jp}) = \text{diag}(\boldsymbol{\sigma}_j)$  are often used for simplicity.
- This is a richer model than the traditional “bag of words” approach.
- The probabilistic model is learned offline beforehand.

## Application: Aggregation of visual words (3/4)

After a few calculations (exercise), we obtain  $\varphi_{\theta}(\mathbf{x}_1, \dots, \mathbf{x}_n) =$

$$[\varphi_{\pi_1}(\mathbf{X}), \dots, \varphi_{\pi_p}(\mathbf{X}), \varphi_{\mu_1}(\mathbf{X})^{\top}, \dots, \varphi_{\mu_p}(\mathbf{X})^{\top}, \varphi_{\sigma_1}(\mathbf{X})^{\top}, \dots, \varphi_{\sigma_p}(\mathbf{X})^{\top}]^{\top},$$

with

$$\varphi_{\mu_j}(\mathbf{X}) = \frac{1}{n\sqrt{\pi_j}} \sum_{i=1}^n \gamma_{ij}(\mathbf{x}_i - \mu_j)/\sigma_j$$

$$\varphi_{\sigma_j}(\mathbf{X}) = \frac{1}{n\sqrt{2\pi_j}} \sum_{i=1}^n \gamma_{ij} [(\mathbf{x}_i - \mu_j)^2/\sigma_j^2 - 1],$$

where with an abuse of notation, the division between two vectors is meant elementwise and the scalars  $\gamma_{ij}$  can be interpreted as the **soft-assignment** of word  $i$  to component  $j$ :

$$\gamma_{ij} = \frac{\pi_j \mathcal{N}(\mathbf{x}_i; \mu_j, \sigma_j)}{\sum_{l=1}^k \pi_l \mathcal{N}(\mathbf{x}_i; \mu_l, \sigma_l)}.$$

## Application: Aggregation of visual words (4/4)

Finally, we also have the following interpretation of encoding first and second-order statistics:

$$\varphi_{\mu_j}(\mathbf{X}) = \frac{\gamma_j}{\sqrt{\pi_j}}(\hat{\mu}_j - \mu_j)/\sigma_j$$
$$\varphi_{\sigma_j}(\mathbf{X}) = \frac{\gamma_j}{\sqrt{2\pi_j}}(\hat{\sigma}_j^2 - \sigma_j^2)/\sigma_j^2,$$

with

$$\gamma_j = \sum_{i=1}^n \gamma_{ij} \quad \text{and} \quad \hat{\mu}_j = \frac{1}{\gamma_j} \sum_{i=1}^n \gamma_{ij} \mathbf{x}_i \quad \text{and} \quad \hat{\sigma}_j^2 = \frac{1}{\gamma_j} \sum_{i=1}^n \gamma_{ij} (\mathbf{x}_i - \mu_j)^2.$$

The component  $\varphi_{\pi}(\mathcal{X})$  is often dropped due to its negligible contribution in practice, and the resulting representation is of dimension  $2kp$  where  $p$  is the dimension of the  $\mathbf{x}_i$ 's.

# Outline

## 4 The Kernel Jungle

- Mercer kernels and shift-invariant kernels
- **Kernels for generative models**
  - Fisher kernel
  - **Mutual information kernels**
  - Marginalized kernels
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs

# Mutual information kernels

## Definition

- Chose a prior  $w(d\theta)$  on the measurable set  $\Theta$ .
- Form the kernel (Seeger, 2002):

$$K(\mathbf{x}, \mathbf{x}') = \int_{\theta \in \Theta} P_{\theta}(\mathbf{x}) P_{\theta}(\mathbf{x}') w(d\theta) .$$

- **No explicit computation** of a finite-dimensional feature vector.
- $K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{L_2(w)}$  with

$$\varphi(\mathbf{x}) = (P_{\theta}(\mathbf{x}))_{\theta \in \Theta} .$$

## Example: coin toss

- Let  $P_\theta(X = 1) = \theta$  and  $P_\theta(X = 0) = 1 - \theta$  a model for random coin toss, with  $\theta \in [0, 1]$ .
- Let  $d\theta$  be the Lebesgue measure on  $[0, 1]$
- The mutual information kernel between  $\mathbf{x} = 001$  and  $\mathbf{x}' = 1010$  is:

$$\begin{cases} P_\theta(\mathbf{x}) &= \theta(1-\theta)^2, \\ P_\theta(\mathbf{x}') &= \theta^2(1-\theta)^2, \end{cases}$$

$$K(\mathbf{x}, \mathbf{x}') = \int_0^1 \theta^3 (1-\theta)^4 d\theta = \frac{3!4!}{8!} = \frac{1}{280}.$$

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
    - Fisher kernel
    - Mutual information kernels
    - Marginalized kernels
  - Kernels for biological sequences
  - Kernels for graphs
  - Kernels on graphs

# Marginalized kernels

## Definition

- For any **observed data**  $\mathbf{x} \in \mathcal{X}$ , let a **latent variable**  $\mathbf{y} \in \mathcal{Y}$  be associated probabilistically through a **conditional probability**  $P_{\mathbf{x}}(d\mathbf{y})$ .
- Let  $K_{\mathcal{Z}}$  be a **kernel for the complete data**  $\mathbf{z} = (\mathbf{x}, \mathbf{y})$
- Then the following kernel is a valid kernel on  $\mathcal{X}$ , called a **marginalized kernel** (Tsuda et al., 2002):

$$\begin{aligned} K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') &:= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') \\ &= \int \int K_{\mathcal{Z}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) P_{\mathbf{x}}(d\mathbf{y}) P_{\mathbf{x}'}(d\mathbf{y}') . \end{aligned}$$

## Marginalized kernels: proof of positive definiteness

- $K_{\mathcal{Z}}$  is p.d. on  $\mathcal{Z}$ . Therefore there exists a Hilbert space  $\mathcal{H}$  and  $\Phi_{\mathcal{Z}} : \mathcal{Z} \rightarrow \mathcal{H}$  such that:

$$K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') = \langle \Phi_{\mathcal{Z}}(\mathbf{z}), \Phi_{\mathcal{Z}}(\mathbf{z}') \rangle_{\mathcal{H}} .$$

- Marginalizing therefore gives:

$$\begin{aligned} K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') \\ &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} \langle \Phi_{\mathcal{Z}}(\mathbf{z}), \Phi_{\mathcal{Z}}(\mathbf{z}') \rangle_{\mathcal{H}} \\ &= \langle E_{P_{\mathbf{x}}(d\mathbf{y})} \Phi_{\mathcal{Z}}(\mathbf{z}), E_{P_{\mathbf{x}'}(d\mathbf{y}')} \Phi_{\mathcal{Z}}(\mathbf{z}') \rangle_{\mathcal{H}} , \end{aligned}$$

therefore  $K_{\mathcal{X}}$  is p.d. on  $\mathcal{X}$ .  $\square$

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - **Kernels for biological sequences**
  - Kernels for graphs
  - Kernels on graphs

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - **Kernels for biological sequences**
    - Motivations and history of genomics
    - Kernels derived from large feature spaces
    - Kernels derived from generative models
    - Kernels derived from a similarity measure
    - Application to remote homology detection
  - Kernels for graphs
  - Kernels on graphs

# Short history of genomics



**1866** : Laws of heredity (Mendel)

1909 : Morgan and the drosophilists

1944 : DNA supports heredity (Avery)

**1953** : Structure of DNA (Crick and Watson)

1966 : Genetic code (Nirenberg)

1960-70 : Genetic engineering

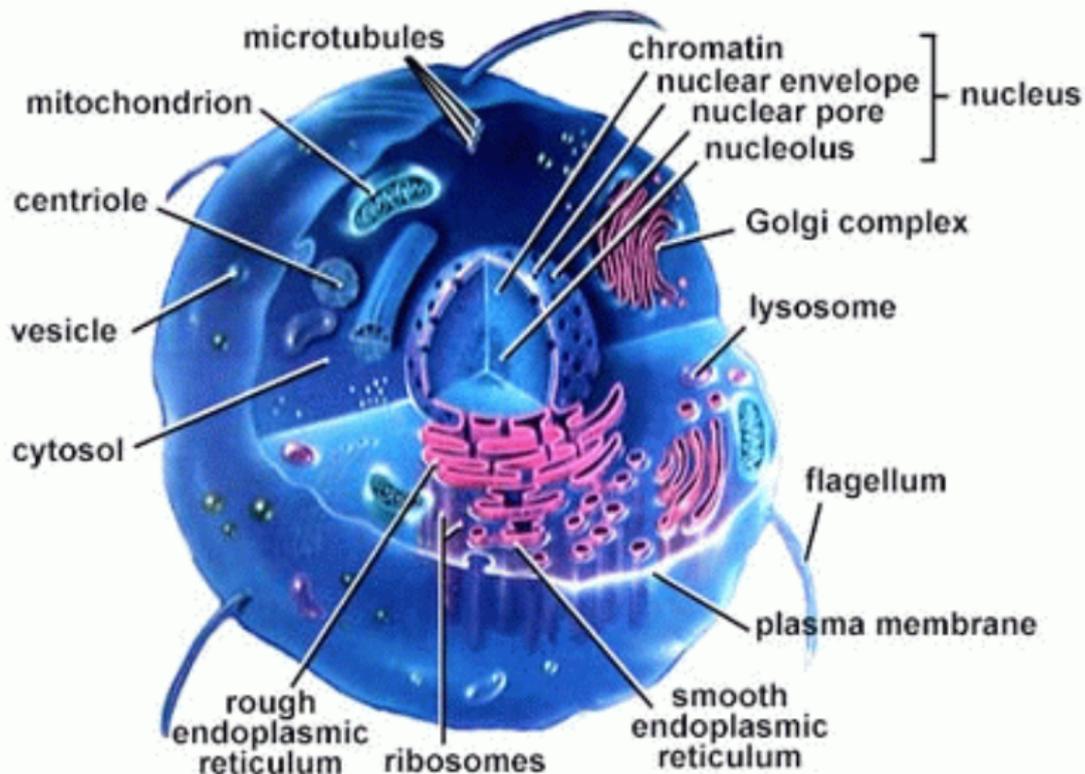
1977 : Method for sequencing (Sanger)

1982 : Creation of Genbank

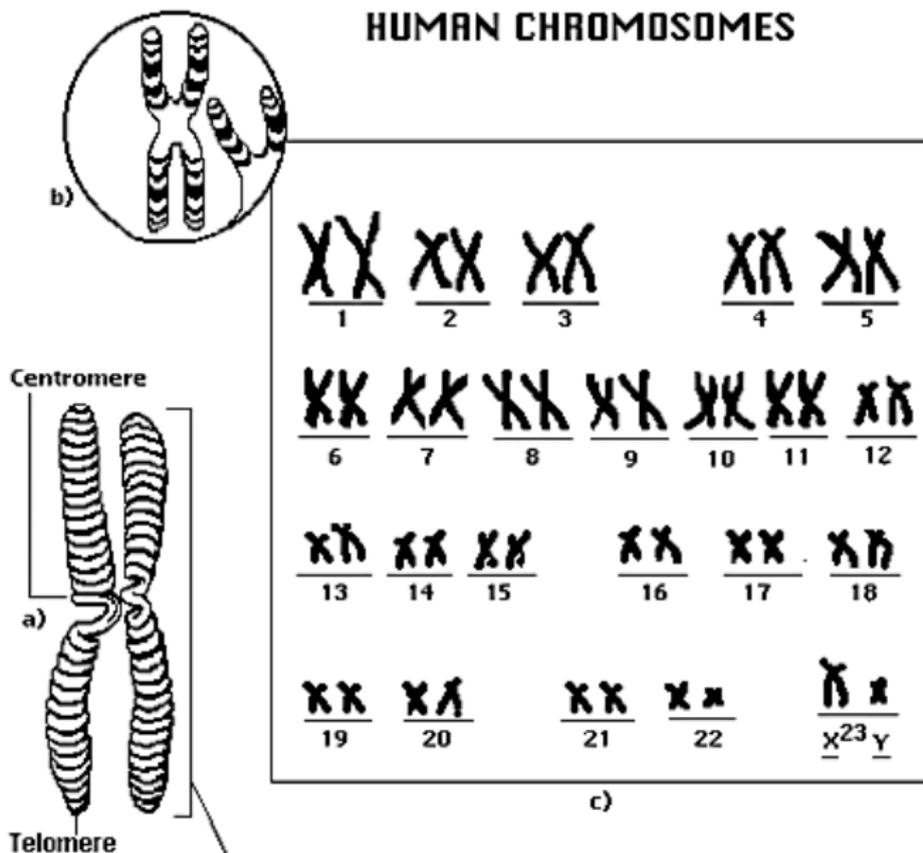
1990 : Human genome project launched

**2003** : Human genome project completed

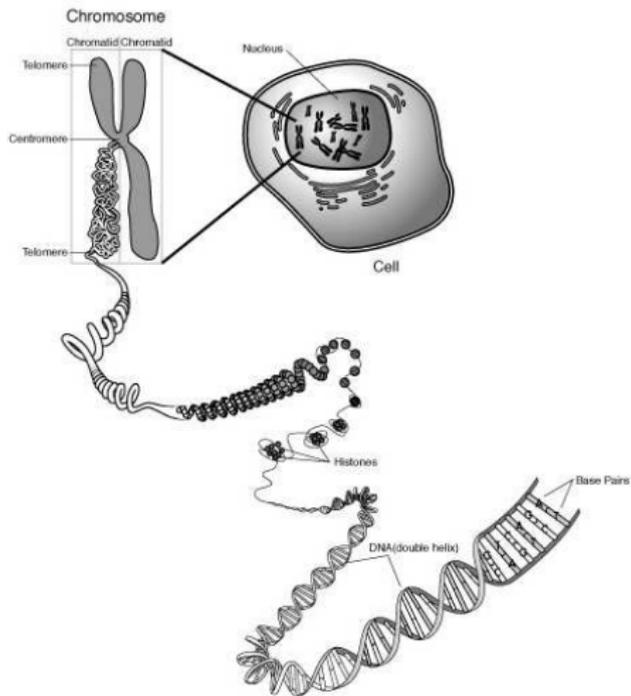
# A cell



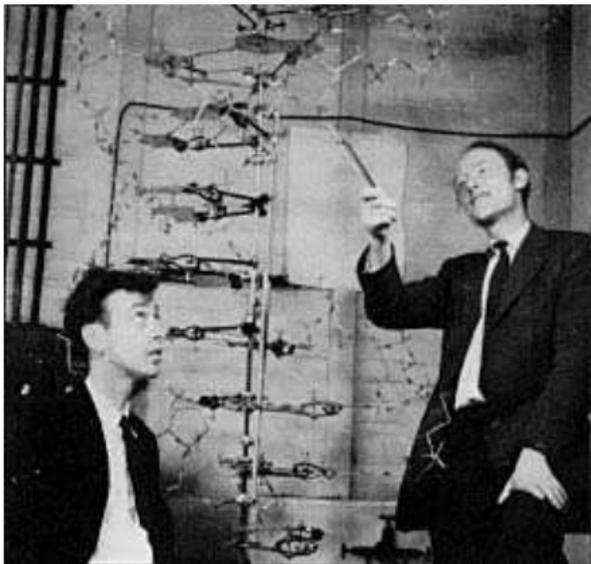
# Chromosomes



# Chromosomes and DNA

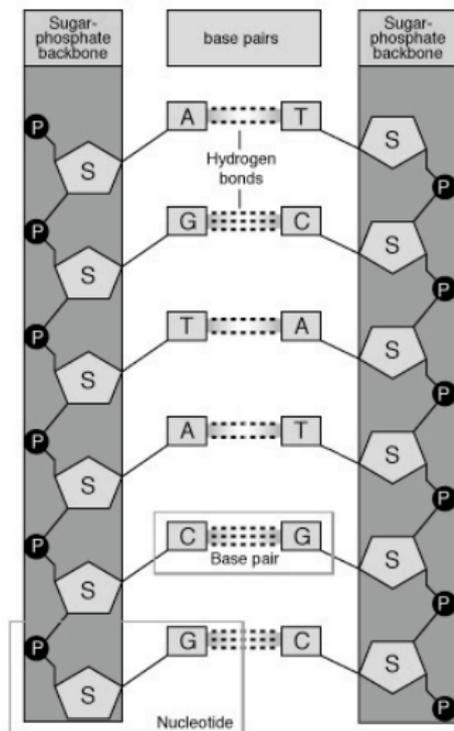
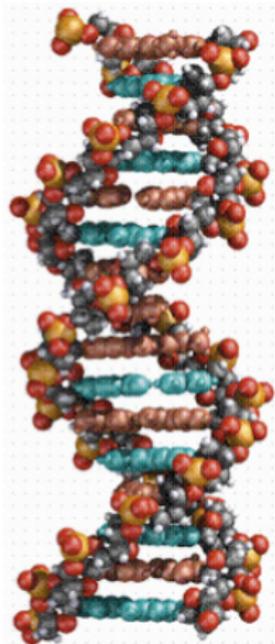


# Structure of DNA

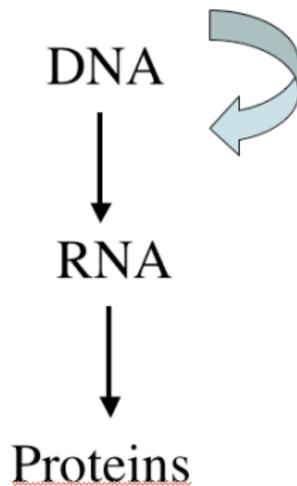
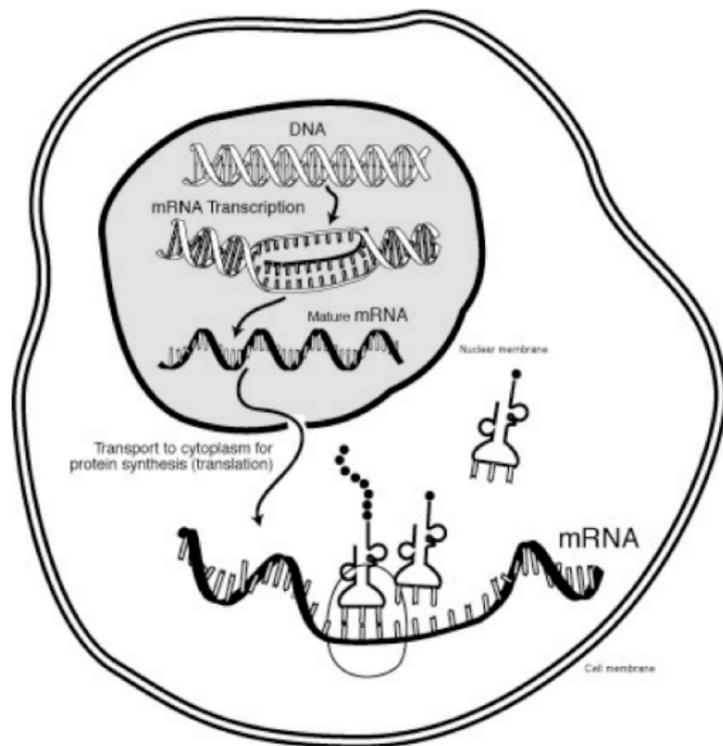


“We wish to suggest a structure for the salt of desoxyribose nucleic acid (D.N.A.). This structure have novel features which are of considerable biological interest” (Watson and Crick, 1953)

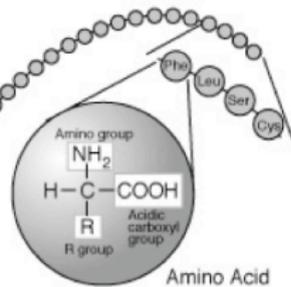
# The double helix



# Central dogma



# Proteins



# Genetic code

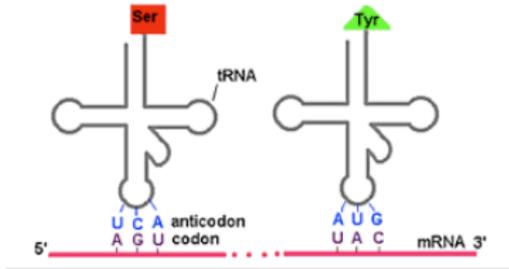
DNA = 4 letters (ATCG)



RNA = 4 letters (AUCG)



Protein = 20 letters (amino acids)



		2nd base in codon					
		U	C	A	G		
1st base in codon	U	Phe Phe Leu Leu	Ser Ser Ser Ser	Tyr Tyr STOP STOP	Cys Cys STOP Trp	U C A G	3rd base in codon
	C	Leu Leu Leu	Pro Pro Pro	His His Gln	Arg Arg Arg	U C A G	
	A	Ile Ile Ile Met	Thr Thr Thr	Asn Asn Lys	Ser Ser Arg Arg	U C A G	
	G	Val Val Val Val	Ala Ala Ala Ala	Asp Asp Glu Glu	Gly Gly Gly Gly	U C A G	

The Genetic Code

1 amino acid  
=  
3 nucleotides

# Human genome project

- Goal : sequence the 3,000,000,000 bases of the human genome
- Consortium with 20 labs, 6 countries
- Cost : about 3,000,000,000 USD



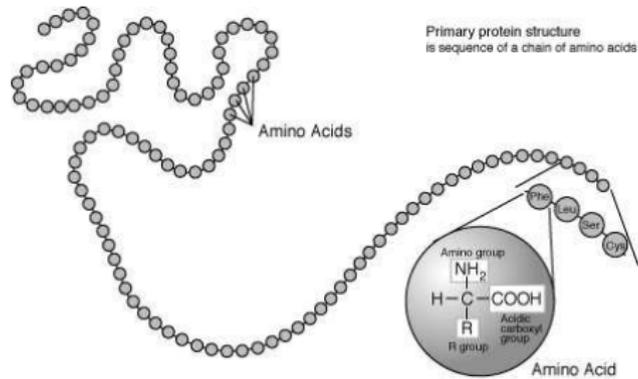
## 2003: End of genomics era



### Findings

- About 25,000 genes only (representing 1.2% of the genome).
- Automatic gene finding with graphical models.
- 97% of the genome is considered “junk DNA”.
- Superposition of a variety of signals (many to be discovered).

# Protein sequence



**A** : Alanine

**F** : Phenylalanine

**E** : Glutamic acid

**T** : Threonine

**H** : Histidine

**I** : Isoleucine

**D** : Aspartic acid

**V** : Valine

**P** : Proline

**K** : Lysine

**C** : Cysteine

**Y** : Tyrosine

**S** : Serine

**G** : Glycine

**L** : Leucine

**M** : Methionine

**R** : Arginine

**N** : Asparagine

**W** : Tryptophane

**Q** : Glutamine

# Challenges with protein sequences

- A protein sequences can be seen as a **variable-length sequence** over the **20-letter alphabet** of amino-acids, e.g., insuline:  
FVNQHLCGSHLVEALYLVCGERGFFYTPKA
- These sequences are produced at a fast rate (result of the **sequencing programs**)
- Need for algorithms to **compare, classify, analyze** these sequences
- Applications: classification into **functional or structural** classes, prediction of **cellular localization** and **interactions**, ...

# Example: supervised sequence classification

## Data (training)

- Secreted proteins:

MASKATLLLAFTLLFATCIARHQQRQQQNQCQLQNIEA...

MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...

MALHTVLIMLSLLPMLEAQNPEHANITIGEPITNETLGWL...

...

- Non-secreted proteins:

MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVN LGVG...

MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...

MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..

...

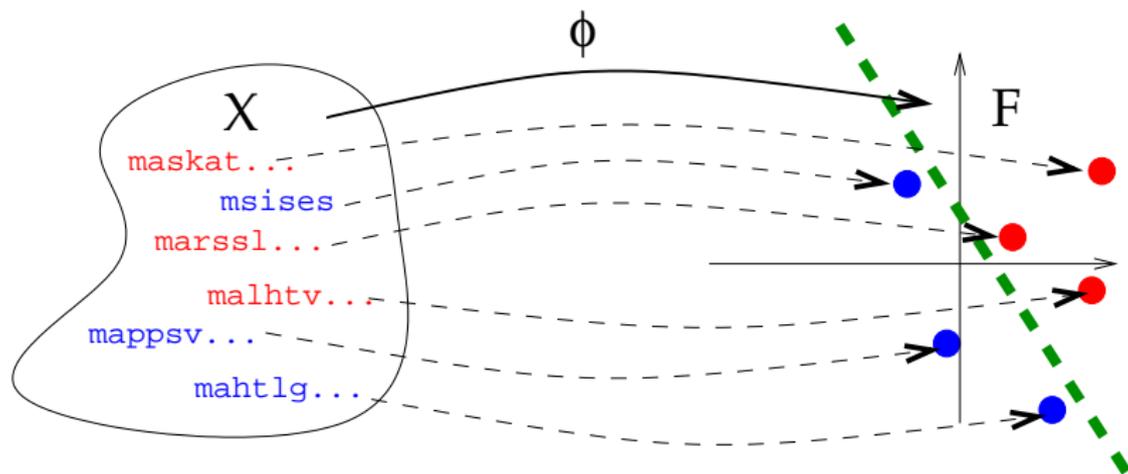
## Goal

- Build a **classifier** to **predict** whether new proteins are secreted or not.

# Supervised classification with vector embedding

## The idea

- Map each string  $\mathbf{x} \in \mathcal{X}$  to a **vector**  $\Phi(\mathbf{x}) \in \mathcal{F}$ .
- Train a **classifier for vectors** on the images  $\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_n)$  of the training set (nearest neighbor, linear perceptron, logistic regression, support vector machine...)



# Kernels for protein sequences

- **Kernel methods** have been widely investigated since Jaakkola et al.'s seminal paper (1998).
- What is a **good kernel**?
  - it should be **mathematically valid** (symmetric, p.d. or c.p.d.)
  - **fast to compute**
  - **adapted to the problem** (give good performances)

# Kernel engineering for protein sequences

- Define a (possibly high-dimensional) **feature space** of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels

# Kernel engineering for protein sequences

- Define a (possibly high-dimensional) **feature space** of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels
- Derive a kernel from a **generative model**
  - Fisher kernel
  - Mutual information kernel
  - Marginalized kernel

# Kernel engineering for protein sequences

- Define a (possibly high-dimensional) **feature space** of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels
- Derive a kernel from a **generative model**
  - Fisher kernel
  - Mutual information kernel
  - Marginalized kernel
- Derive a kernel from a **similarity measure**
  - Local alignment kernel

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - **Kernels for biological sequences**
    - Motivations and history of genomics
    - **Kernels derived from large feature spaces**
    - Kernels derived from generative models
    - Kernels derived from a similarity measure
    - Application to remote homology detection
  - Kernels for graphs
  - Kernels on graphs

# Vector embedding for strings

## The idea

Represent each sequence  $\mathbf{x}$  by a **fixed-length numerical vector**  $\Phi(\mathbf{x}) \in \mathbb{R}^n$ . How to perform this embedding?

# Vector embedding for strings

## The idea

Represent each sequence  $\mathbf{x}$  by a **fixed-length numerical vector**  $\Phi(\mathbf{x}) \in \mathbb{R}^n$ . How to perform this embedding?

## Physico-chemical kernel

Extract **relevant features**, such as:

- length of the sequence
- **time series analysis of numerical physico-chemical properties** of amino-acids along the sequence (e.g., polarity, hydrophobicity), using for example:
  - Fourier transforms (Wang et al., 2004)
  - Autocorrelation functions (Zhang et al., 2003)

$$r_j = \frac{1}{n-j} \sum_{i=1}^{n-j} h_i h_{i+j}$$

# Substring indexation

## The approach

Alternatively, index the feature space by fixed-length strings, i.e.,

$$\Phi(\mathbf{x}) = (\Phi_u(\mathbf{x}))_{u \in \mathcal{A}^k}$$

where  $\Phi_u(\mathbf{x})$  can be:

- the number of occurrences of  $u$  in  $\mathbf{x}$  (without gaps) : **spectrum kernel** (Leslie et al., 2002)
- the number of occurrences of  $u$  in  $\mathbf{x}$  up to  $m$  mismatches (without gaps) : **mismatch kernel** (Leslie et al., 2004)
- the number of occurrences of  $u$  in  $\mathbf{x}$  allowing gaps, with a weight decaying exponentially with the number of gaps : **substring kernel** (Lohdi et al., 2002)

## Example: spectrum kernel (1/2)

### Kernel definition

- The 3-spectrum of

$\mathbf{x} = \text{CGGS LIAMM WFGV}$

is:

$(\text{CGG}, \text{GGS}, \text{GSL}, \text{SLI}, \text{LIA}, \text{IAM}, \text{AMM}, \text{MMW}, \text{MWF}, \text{WFG}, \text{FGV})$ .

- Let  $\Phi_u(\mathbf{x})$  denote the number of occurrences of  $u$  in  $\mathbf{x}$ . The  $k$ -spectrum kernel is:

$$K(\mathbf{x}, \mathbf{x}') := \sum_{u \in \mathcal{A}^k} \Phi_u(\mathbf{x}) \Phi_u(\mathbf{x}') .$$

## Example: spectrum kernel (2/2)

### Implementation

- The computation of the kernel is formally a sum over  $|\mathcal{A}|^k$  terms, but at most  $|\mathbf{x}| - k + 1$  terms are non-zero in  $\Phi(\mathbf{x}) \implies$   
**Computation in  $O(|\mathbf{x}| + |\mathbf{x}'|)$**  with pre-indexation of the strings.
- Fast classification of a sequence  $\mathbf{x}$  in  $O(|\mathbf{x}|)$ :

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_u w_u \Phi_u(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|-k+1} w_{x_i \dots x_{i+k-1}}.$$

### Remarks

- Work with any string (natural language, time series...)
- **Fast and scalable**, a good default method for string classification.
- Variants allow matching of  $k$ -mers up to  $m$  **mismatches**.

## Example 2: Substring kernel (1/11)

### Definition

- For  $1 \leq k \leq n \in \mathbb{N}$ , we denote by  $\mathcal{I}(k, n)$  the set of **sequences of indices**  $\mathbf{i} = (i_1, \dots, i_k)$ , with  $1 \leq i_1 < i_2 < \dots < i_k \leq n$ .
- For a string  $\mathbf{x} = x_1 \dots x_n \in \mathcal{X}$  of length  $n$ , for a sequence of indices  $\mathbf{i} \in \mathcal{I}(k, n)$ , we define a **substring** as:

$$\mathbf{x}(\mathbf{i}) := x_{i_1} x_{i_2} \dots x_{i_k}.$$

- The **length** of the substring is:

$$l(\mathbf{i}) = i_k - i_1 + 1.$$

## Example 2: Substring kernel (2/11)

### Example

ABRACADABRA

- $\mathbf{i} = (3, 4, 7, 8, 10)$
- $\mathbf{x}(\mathbf{i}) = \text{RADAR}$
- $l(\mathbf{i}) = 10 - 3 + 1 = 8$

## Example 2: Substring kernel (3/11)

### The kernel

- Let  $k \in \mathbb{N}$  and  $\lambda \in \mathbb{R}^+$  fixed. For all  $\mathbf{u} \in \mathcal{A}^k$ , let  $\Phi_{\mathbf{u}} : \mathcal{X} \rightarrow \mathbb{R}$  be defined by:

$$\forall \mathbf{x} \in \mathcal{X}, \quad \Phi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i} \in \mathcal{I}(k, |\mathbf{x}|): \mathbf{x}(\mathbf{i}) = \mathbf{u}} \lambda^{|\mathbf{i}|}.$$

- The **substring kernel** is the p.d. kernel defined by:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K_{k, \lambda}(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{u} \in \mathcal{A}^k} \Phi_{\mathbf{u}}(\mathbf{x}) \Phi_{\mathbf{u}}(\mathbf{x}').$$

## Example 2: Substring kernel (4/11)

### Example

u	ca	ct	at	ba	bt	cr	ar	br
$\Phi_u(\text{cat})$	$\lambda^2$	$\lambda^3$	$\lambda^2$	0	0	0	0	0
$\Phi_u(\text{car})$	$\lambda^2$	0	0	0	0	$\lambda^3$	$\lambda^2$	0
$\Phi_u(\text{bat})$	0	0	$\lambda^2$	$\lambda^2$	$\lambda^3$	0	0	0
$\Phi_u(\text{bar})$	0	0	0	$\lambda^2$	0	0	$\lambda^2$	$\lambda^3$

$$\begin{cases} K(\text{cat}, \text{cat}) = K(\text{car}, \text{car}) = 2\lambda^4 + \lambda^6 \\ K(\text{cat}, \text{car}) = \lambda^4 \\ K(\text{cat}, \text{bar}) = 0 \end{cases}$$

## Example 2: Substring kernel (5/11)

### Kernel computation

- We need to compute, for any pair  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , the kernel:

$$\begin{aligned} K_{n,\lambda}(\mathbf{x}, \mathbf{x}') &= \sum_{\mathbf{u} \in \mathcal{A}^k} \Phi_{\mathbf{u}}(\mathbf{x}) \Phi_{\mathbf{u}}(\mathbf{x}') \\ &= \sum_{\mathbf{u} \in \mathcal{A}^k} \sum_{\mathbf{i}: \mathbf{x}(\mathbf{i})=\mathbf{u}} \sum_{\mathbf{i}': \mathbf{x}'(\mathbf{i}')=\mathbf{u}} \lambda^{l(\mathbf{i})+l(\mathbf{i}')}. \end{aligned}$$

- Enumerating the substrings is **too slow** (of order  $|\mathbf{x}|^k$ ).

## Example 2: Substring kernel (6/11)

### Kernel computation (cont.)

- For  $\mathbf{u} \in \mathcal{A}^k$  remember that:

$$\Phi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i}: \mathbf{x}(\mathbf{i}) = \mathbf{u}} \lambda^{i_n - i_1 + 1}.$$

- Let now:

$$\Psi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i}: \mathbf{x}(\mathbf{i}) = \mathbf{u}} \lambda^{|\mathbf{x}| - i_1 + 1}.$$

## Example 2: Substring kernel (7/11)

### Kernel computation (cont.)

Let us note  $\mathbf{x}(1, j) = x_1 \dots x_j$ . A simple rewriting shows that, if we note  $a \in \mathcal{A}$  the last letter of  $\mathbf{u}$  ( $\mathbf{u} = \mathbf{v}a$ ):

$$\Phi_{\mathbf{v}a}(\mathbf{x}) = \sum_{j \in [1, |\mathbf{x}|]: x_j = a} \Psi_{\mathbf{v}}(\mathbf{x}(1, j-1)) \lambda,$$

and

$$\Psi_{\mathbf{v}a}(\mathbf{x}) = \sum_{j \in [1, |\mathbf{x}|]: x_j = a} \Psi_{\mathbf{v}}(\mathbf{x}(1, j-1)) \lambda^{|\mathbf{x}|-j+1}.$$

## Example 2: Substring kernel (8/11)

### Kernel computation (cont.)

Moreover we observe that if the string is of the form  $\mathbf{x}a$  (i.e., the last letter is  $a \in \mathcal{A}$ ), then:

- If the last letter of  $\mathbf{u}$  is not  $a$ :

$$\begin{cases} \Phi_{\mathbf{u}}(\mathbf{x}a) &= \Phi_{\mathbf{u}}(\mathbf{x}) , \\ \Psi_{\mathbf{u}}(\mathbf{x}a) &= \lambda \Psi_{\mathbf{u}}(\mathbf{x}) . \end{cases}$$

- If the last letter of  $\mathbf{u}$  is  $a$  (i.e.,  $\mathbf{u} = \mathbf{v}a$  with  $\mathbf{v} \in \mathcal{A}^{n-1}$ ):

$$\begin{cases} \Phi_{\mathbf{v}a}(\mathbf{x}a) &= \Phi_{\mathbf{v}a}(\mathbf{x}) + \lambda \Psi_{\mathbf{v}}(\mathbf{x}) , \\ \Psi_{\mathbf{v}a}(\mathbf{x}a) &= \lambda \Psi_{\mathbf{v}a}(\mathbf{x}) + \lambda \Psi_{\mathbf{v}}(\mathbf{x}) . \end{cases}$$

## Example 2: Substring kernel (9/11)

### Kernel computation (cont.)

Let us now show how the function:

$$B_n(\mathbf{x}, \mathbf{x}') := \sum_{\mathbf{u} \in \mathcal{A}^n} \Psi_{\mathbf{u}}(\mathbf{x}) \Psi_{\mathbf{u}}(\mathbf{x}')$$

and the kernel:

$$K_n(\mathbf{x}, \mathbf{x}') := \sum_{\mathbf{u} \in \mathcal{A}^n} \Phi_{\mathbf{u}}(\mathbf{x}) \Phi_{\mathbf{u}}(\mathbf{x}')$$

can be computed recursively. We note that:

$$\begin{cases} B_0(\mathbf{x}, \mathbf{x}') = K_0(\mathbf{x}, \mathbf{x}') = 0 & \text{for all } \mathbf{x}, \mathbf{x}' \\ B_k(\mathbf{x}, \mathbf{x}') = K_k(\mathbf{x}, \mathbf{x}') = 0 & \text{if } \min(|\mathbf{x}|, |\mathbf{x}'|) < k \end{cases}$$

## Example 2: Substring kernel (10/11)

### Recursive computation of $B_n$

$$\begin{aligned} B_n(\mathbf{x}a, \mathbf{x}') &= \sum_{\mathbf{u} \in \mathcal{A}^n} \Psi_{\mathbf{u}}(\mathbf{x}a) \Psi_{\mathbf{u}}(\mathbf{x}') \\ &= \lambda \sum_{\mathbf{u} \in \mathcal{A}^n} \Psi_{\mathbf{u}}(\mathbf{x}) \Psi_{\mathbf{u}}(\mathbf{x}') + \lambda \sum_{\mathbf{v} \in \mathcal{A}^{n-1}} \Psi_{\mathbf{v}}(\mathbf{x}) \Psi_{\mathbf{v}a}(\mathbf{x}') \\ &= \lambda B_n(\mathbf{x}, \mathbf{x}') + \\ &\quad \lambda \sum_{\mathbf{v} \in \mathcal{A}^{n-1}} \Psi_{\mathbf{v}}(\mathbf{x}) \left( \sum_{j \in [1, |\mathbf{x}'|]: \mathbf{x}'_j = a} \Psi_{\mathbf{v}}(\mathbf{x}'(1, j-1)) \lambda^{|\mathbf{x}'| - j + 1} \right) \\ &= \lambda B_n(\mathbf{x}, \mathbf{x}') + \sum_{j \in [1, |\mathbf{x}'|]: \mathbf{x}'_j = a} B_{n-1}(\mathbf{x}, \mathbf{x}'(1, j-1)) \lambda^{|\mathbf{x}'| - j + 2} \end{aligned}$$

## Example 2: Substring kernel (10/11)

### Recursive computation of $K_n$

$$\begin{aligned} & K_n(\mathbf{x}a, \mathbf{x}') \\ &= \sum_{\mathbf{u} \in \mathcal{A}^n} \Phi_{\mathbf{u}}(\mathbf{x}a) \Phi_{\mathbf{u}}(\mathbf{x}') \\ &= \sum_{\mathbf{u} \in \mathcal{A}^n} \Phi_{\mathbf{u}}(\mathbf{x}) \Phi_{\mathbf{u}}(\mathbf{x}') + \lambda \sum_{\mathbf{v} \in \mathcal{A}^{n-1}} \Psi_{\mathbf{v}}(\mathbf{x}) \Phi_{\mathbf{v}a}(\mathbf{x}') \\ &= K_n(\mathbf{x}, \mathbf{x}') + \\ & \quad \lambda \sum_{\mathbf{v} \in \mathcal{A}^{n-1}} \Psi_{\mathbf{v}}(\mathbf{x}) \left( \sum_{j \in [1, |\mathbf{x}'|]: \mathbf{x}'_j = a} \Psi_{\mathbf{v}}(\mathbf{x}'(1, j-1)) \lambda \right) \\ &= \lambda K_n(\mathbf{x}, \mathbf{x}') + \lambda^2 \sum_{j \in [1, |\mathbf{x}'|]: \mathbf{x}'_j = a} B_{n-1}(\mathbf{x}, \mathbf{x}'(1, j-1)) \end{aligned}$$

## Summary: Substring indexation

- Implementation in  $O(|\mathbf{x}| + |\mathbf{x}'|)$  in memory and time for the spectrum and mismatch kernels (with suffix trees)
- Implementation in  $O(|\mathbf{x}| \times |\mathbf{x}'|)$  in memory and time for the substring kernels
- The feature space has high dimension ( $|\mathcal{A}|^k$ ), so learning requires **regularized methods** (such as SVM)

# Dictionary-based indexation

## The approach

- Chose a **dictionary** of sequences  $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$
- Chose a **measure of similarity**  $s(\mathbf{x}, \mathbf{x}')$
- Define the mapping  $\Phi_{\mathcal{D}}(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_i))_{\mathbf{x}_i \in \mathcal{D}}$

# Dictionary-based indexation

## The approach

- Chose a **dictionary** of sequences  $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$
- Chose a **measure of similarity**  $s(\mathbf{x}, \mathbf{x}')$
- Define the mapping  $\Phi_{\mathcal{D}}(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_i))_{\mathbf{x}_i \in \mathcal{D}}$

## Examples

This includes:

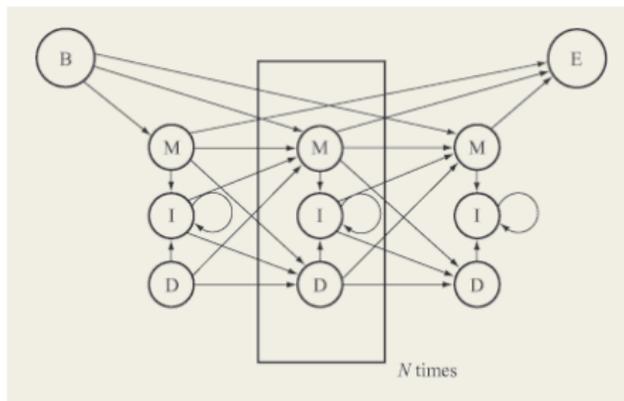
- **Motif kernels** (Logan et al., 2001): the dictionary is a library of motifs, the similarity function is a matching function
- **Pairwise kernel** (Liao & Noble, 2003): the dictionary is the training set, the similarity is a classical measure of similarity between sequences.

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - **Kernels for biological sequences**
    - Motivations and history of genomics
    - Kernels derived from large feature spaces
    - **Kernels derived from generative models**
    - Kernels derived from a similarity measure
    - Application to remote homology detection
  - Kernels for graphs
  - Kernels on graphs

# Probabilistic models for sequences

**Probabilistic modeling** of biological sequences is older than kernel designs. Important models include **HMM** for protein sequences, **SCFG** for RNA sequences.



Recall: parametric model

A **model** is a family of distribution

$$\{P_{\theta}, \theta \in \Theta \subset \mathbb{R}^m\} \subset \mathcal{M}_1^+(\mathcal{X})$$

# Context-tree model

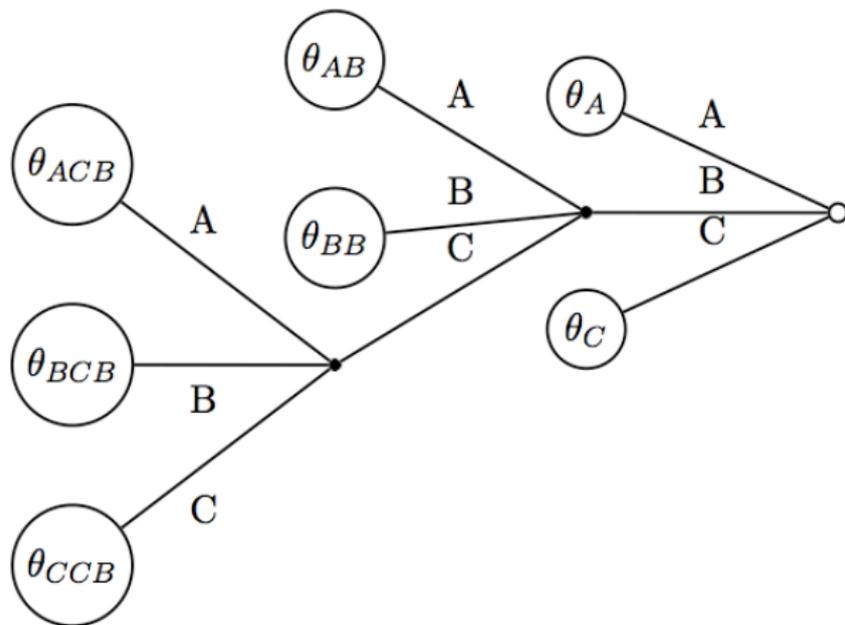
## Definition

A context-tree model is a **variable-memory Markov chain**:

$$P_{\mathcal{D},\theta}(\mathbf{x}) = P_{\mathcal{D},\theta}(x_1 \dots x_D) \prod_{i=D+1}^n P_{\mathcal{D},\theta}(x_i | x_{i-D} \dots x_{i-1})$$

- $\mathcal{D}$  is a suffix tree
- $\theta \in \Sigma^{\mathcal{D}}$  is a set of conditional probabilities (multinomials)

## Context-tree model: example



$$P(AABACBACC) = P(AAB)\theta_{AB}(A)\theta_A(C)\theta_C(B)\theta_{ACB}(A)\theta_A(C)\theta_C(A).$$

# The context-tree kernel

## Theorem (Cuturi et al., 2005)

- For particular choices of priors, the context-tree kernel:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{\mathcal{D}} \int_{\theta \in \Sigma^{\mathcal{D}}} P_{\mathcal{D}, \theta}(\mathbf{x}) P_{\mathcal{D}, \theta}(\mathbf{x}') w(d\theta | \mathcal{D}) \pi(\mathcal{D})$$

can be computed in  $O(|\mathbf{x}| + |\mathbf{x}'|)$  with a variant of the *Context-Tree Weighting algorithm*.

- This is a *valid mutual information kernel*.
- The similarity is related to information-theoretical measure of *mutual information* between strings.

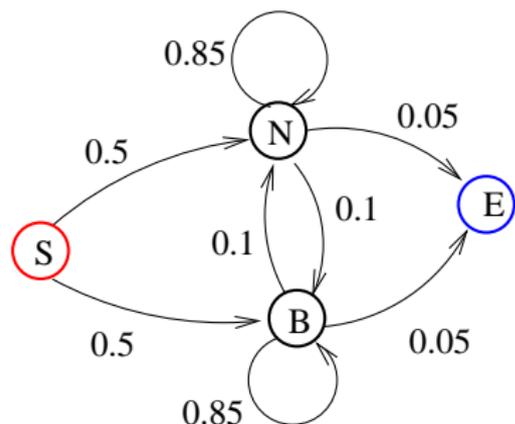
# Marginalized kernels

## Recall: Definition

- For any **observed data**  $\mathbf{x} \in \mathcal{X}$ , let a **latent variable**  $\mathbf{y} \in \mathcal{Y}$  be associated probabilistically through a **conditional probability**  $P_{\mathbf{x}}(d\mathbf{y})$ .
- Let  $K_{\mathcal{Z}}$  be a **kernel for the complete data**  $\mathbf{z} = (\mathbf{x}, \mathbf{y})$
- Then the following kernel is a valid kernel on  $\mathcal{X}$ , called a **marginalized kernel** (Tsuda et al., 2002):

$$\begin{aligned} K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') &:= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') \\ &= \int \int K_{\mathcal{Z}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) P_{\mathbf{x}}(d\mathbf{y}) P_{\mathbf{x}'}(d\mathbf{y}') . \end{aligned}$$

## Example: HMM for normal/biased coin toss



- Normal ( $N$ ) and biased ( $B$ ) coins (not observed)

- Observed output are 0/1 with probabilities:

$$\begin{cases} \pi(0|N) = 1 - \pi(1|N) = 0.5, \\ \pi(0|B) = 1 - \pi(1|B) = 0.8. \end{cases}$$

- Example of realization (complete data):

NNNNBBBBBBBBNNNNNNNNNNBBBBBB  
1001011101111010010111001111011

## 1-spectrum kernel on complete data

- If both  $\mathbf{x} \in \mathcal{A}^*$  and  $\mathbf{y} \in \mathcal{S}^*$  were observed, we might rather use the 1-spectrum kernel on the complete data  $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ :

$$K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') = \sum_{(a,s) \in \mathcal{A} \times \mathcal{S}} n_{a,s}(\mathbf{z}) n_{a,s}(\mathbf{z}'),$$

where  $n_{a,s}(\mathbf{x}, \mathbf{y})$  for  $a = 0, 1$  and  $s = N, B$  is the number of occurrences of  $s$  in  $\mathbf{y}$  which emit  $a$  in  $\mathbf{x}$ .

- Example:

$$\begin{aligned} \mathbf{z} &= 1001011101111010010111001111011, \\ \mathbf{z}' &= 0011010110011111011010111101100101, \end{aligned}$$

$$\begin{aligned} K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') &= n_0(\mathbf{z}) n_0(\mathbf{z}') + n_1(\mathbf{z}) n_1(\mathbf{z}') + n_N(\mathbf{z}) n_N(\mathbf{z}') + n_B(\mathbf{z}) n_B(\mathbf{z}') \\ &= 7 \times 15 + 9 \times 12 + 13 \times 6 + 2 \times 1 = 293. \end{aligned}$$

# 1-spectrum marginalized kernel on observed data

- The marginalized kernel for observed data is:

$$\begin{aligned} K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') &= \sum_{\mathbf{y}, \mathbf{y}' \in \mathcal{S}^*} K_{\mathcal{Z}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}, \mathbf{y}')) P(\mathbf{y}|\mathbf{x}) P(\mathbf{y}'|\mathbf{x}') \\ &= \sum_{(a,s) \in \mathcal{A} \times \mathcal{S}} \Phi_{a,s}(\mathbf{x}) \Phi_{a,s}(\mathbf{x}'), \end{aligned}$$

with

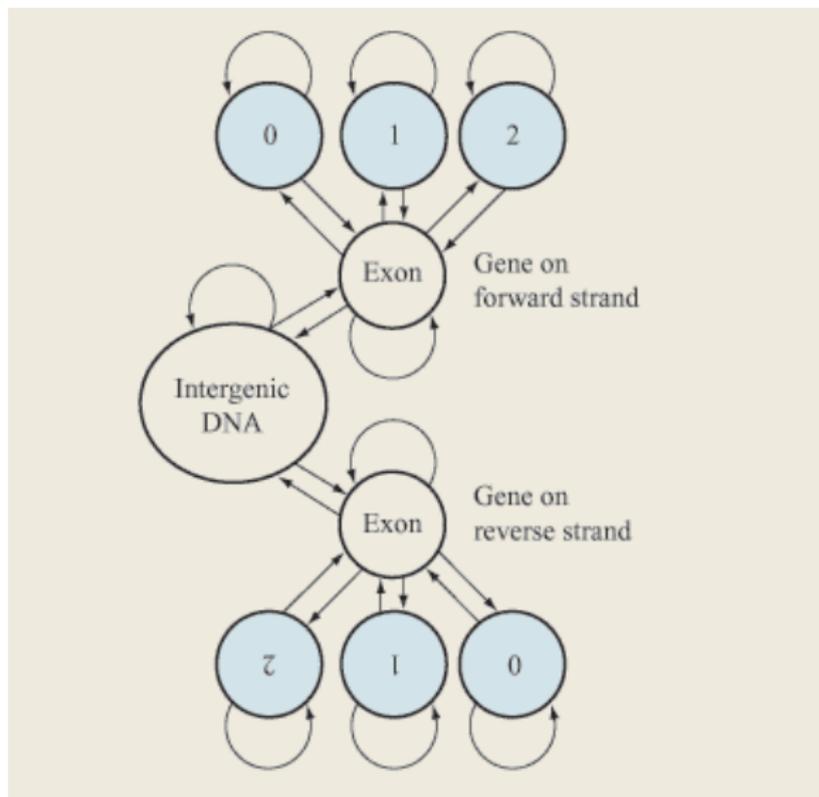
$$\Phi_{a,s}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) n_{a,s}(\mathbf{x}, \mathbf{y})$$

## Computation of the 1-spectrum marginalized kernel

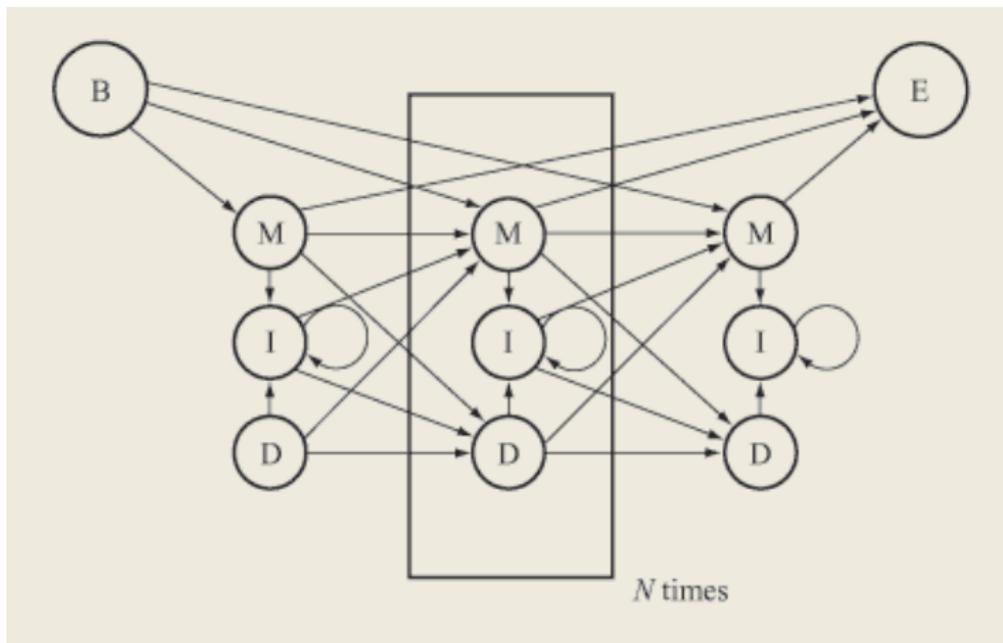
$$\begin{aligned}\Phi_{a,s}(\mathbf{x}) &= \sum_{\mathbf{y} \in \mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) n_{a,s}(\mathbf{x}, \mathbf{y}) \\ &= \sum_{\mathbf{y} \in \mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) \left\{ \sum_{i=1}^n \delta(x_i, a) \delta(y_i, s) \right\} \\ &= \sum_{i=1}^n \delta(x_i, a) \left\{ \sum_{\mathbf{y} \in \mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) \delta(y_i, s) \right\} \\ &= \sum_{i=1}^n \delta(x_i, a) P(y_i = s|\mathbf{x}).\end{aligned}$$

and  $P(y_i = s|\mathbf{x})$  can be computed efficiently by forward-backward algorithm!

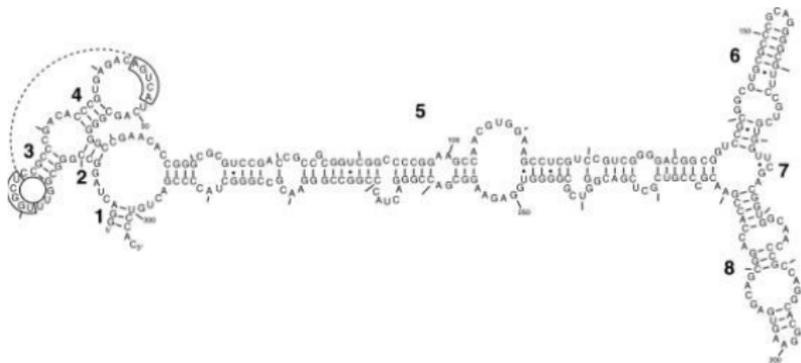
# HMM example (DNA)



# HMM example (protein)



# SCFG for RNA sequences



## SFCG rules

- $S \rightarrow SS$
- $S \rightarrow aSa$
- $S \rightarrow aS$
- $S \rightarrow a$

## Marginalized kernel (Kin et al., 2002)

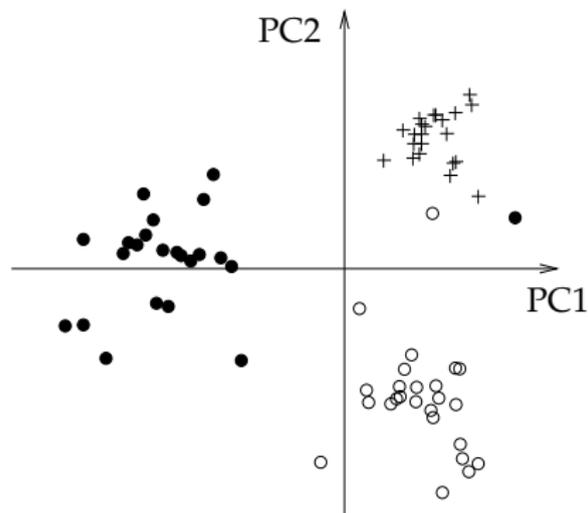
- Feature: number of occurrences of each (base,state) combination
- Marginalization using classical inside/outside algorithm

# Marginalized kernels in practice

## Examples

- Spectrum kernel on the hidden states of a HMM for **protein sequences** (Tsuda et al., 2002)
- Kernels for **RNA sequences** based on SCFG (Kin et al., 2002)
- Kernels for **graphs** based on random walks on graphs (Kashima et al., 2004)
- Kernels for **multiple alignments** based on phylogenetic models (Vert et al., 2006)

## Marginalized kernels: example



A set of 74 human tRNA sequences is analyzed using a kernel for sequences (the second-order marginalized kernel based on SCFG). This set of tRNAs contains three classes, called Ala-AGC (*white circles*), Asn-GTT (*black circles*) and Cys-GCA (*plus symbols*) (from Tsuda et al., 2002).

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - **Kernels for biological sequences**
    - Motivations and history of genomics
    - Kernels derived from large feature spaces
    - Kernels derived from generative models
    - **Kernels derived from a similarity measure**
    - Application to remote homology detection
  - Kernels for graphs
  - Kernels on graphs

# Sequence alignment

## Motivation

How to compare 2 sequences?

$x_1 = \text{CGGSLIAMMWFGV}$

$x_2 = \text{CLIVMMNRLMWFGV}$

Find a good **alignment**:

```
CGGSLIAMM-----WFGV
|...|||||...||||
C-----LIVMMNRLMWFGV
```

## Alignment score

In order to quantify the relevance of an alignment  $\pi$ , define:

- a **substitution matrix**  $S \in \mathbb{R}^{\mathcal{A} \times \mathcal{A}}$
- a **gap penalty** function  $g : \mathbb{N} \rightarrow \mathbb{R}$

Any alignment is then scored as follows

```
CGGSLIAMM-----WFGV
|...|||||...||||
C----LIVMMNRLMWFGV
```

$$s_{S,g}(\pi) = S(C, C) + S(L, L) + S(I, I) + S(A, V) + 2S(M, M) \\ + S(W, W) + S(F, F) + S(G, G) + S(V, V) - g(3) - g(4)$$

# Local alignment kernel

## Smith-Waterman score (Smith and Waterman, 1981)

- The widely-used Smith-Waterman local alignment score is defined by:

$$SW_{S,g}(\mathbf{x}, \mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x}, \mathbf{y})} s_{S,g}(\pi).$$

- It is symmetric, but not positive definite...

# Local alignment kernel

## Smith-Waterman score (Smith and Waterman, 1981)

- The widely-used Smith-Waterman local alignment score is defined by:

$$SW_{S,g}(\mathbf{x}, \mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x}, \mathbf{y})} s_{S,g}(\pi).$$

- It is symmetric, but not positive definite...

## LA kernel (Saigo et al., 2004)

The **local alignment kernel**:

$$K_{LA}^{(\beta)}(\mathbf{x}, \mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x}, \mathbf{y})} \exp(\beta s_{S,g}(\mathbf{x}, \mathbf{y}, \pi)),$$

is **symmetric positive definite**.

## LA kernel is p.d.: proof (1/11)

### Lemma

- If  $K_1$  and  $K_2$  are p.d. kernels, then:

$$K_1 + K_2,$$

$$K_1 K_2, \text{ and}$$

$$cK_1, \text{ for } c \geq 0,$$

are also p.d. kernels

- If  $(K_i)_{i \geq 1}$  is a sequence of p.d. kernels that converges pointwisely to a function  $K$ :

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = \lim_{n \rightarrow \infty} K_n(\mathbf{x}, \mathbf{x}'),$$

then  $K$  is also a p.d. kernel.

## LA kernel is p.d.: proof (2/11)

### Proof of lemma

Let  $A$  and  $B$  be  $n \times n$  positive semidefinite matrices. By diagonalization of  $A$ :

$$A_{i,j} = \sum_{p=1}^n f_p(i)f_p(j)$$

for some vectors  $f_1, \dots, f_n$ . Then, for any  $\alpha \in \mathbb{R}^n$ :

$$\sum_{i,j=1}^n \alpha_i \alpha_j A_{i,j} B_{i,j} = \sum_{p=1}^n \sum_{i,j=1}^n \alpha_i f_p(i) \alpha_j f_p(j) B_{i,j} \geq 0.$$

The matrix  $C_{i,j} = A_{i,j} B_{i,j}$  is therefore p.d. Other properties are obvious from definition.  $\square$

## LA kernel is p.d.: proof (3/11)

### Lemma (direct sum and product of kernels)

Let  $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$ . Let  $K_1$  be a p.d. kernel on  $\mathcal{X}_1$ , and  $K_2$  be a p.d. kernel on  $\mathcal{X}_2$ . Then the following functions are p.d. kernels on  $\mathcal{X}$ :

- the **direct sum**,

$$K((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2)) = K_1(\mathbf{x}_1, \mathbf{y}_1) + K_2(\mathbf{x}_2, \mathbf{y}_2),$$

- The **direct product**:

$$K((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2)) = K_1(\mathbf{x}_1, \mathbf{y}_1) K_2(\mathbf{x}_2, \mathbf{y}_2).$$

## LA kernel is p.d.: proof (4/11)

### Proof of lemma

If  $K_1$  is a p.d. kernel, let  $\Phi_1 : \mathcal{X}_1 \mapsto \mathcal{H}$  be such that:

$$K_1(\mathbf{x}_1, \mathbf{y}_1) = \langle \Phi_1(\mathbf{x}_1), \Phi_1(\mathbf{y}_1) \rangle_{\mathcal{H}}.$$

Let  $\Phi : \mathcal{X}_1 \times \mathcal{X}_2 \rightarrow \mathcal{H}$  be defined by:

$$\Phi((\mathbf{x}_1, \mathbf{x}_2)) = \Phi_1(\mathbf{x}_1).$$

Then for  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$  and  $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2) \in \mathcal{X}$ , we get

$$\langle \Phi((\mathbf{x}_1, \mathbf{x}_2)), \Phi((\mathbf{y}_1, \mathbf{y}_2)) \rangle_{\mathcal{H}} = K_1(\mathbf{x}_1, \mathbf{y}_1),$$

which shows that  $K(\mathbf{x}, \mathbf{y}) := K_1(\mathbf{x}_1, \mathbf{y}_1)$  is p.d. on  $\mathcal{X}_1 \times \mathcal{X}_2$ . The lemma follows from the properties of sums and products of p.d. kernels.  $\square$

## LA kernel is p.d.: proof (5/11)

### Lemma: kernel for sets

Let  $K$  be a p.d. kernel on  $\mathcal{X}$ , and let  $\mathcal{P}(\mathcal{X})$  be the set of **finite subsets** of  $\mathcal{X}$ . Then the function  $K_{\mathcal{P}}$  on  $\mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X})$  defined by:

$$\forall A, B \in \mathcal{P}(\mathcal{X}), \quad K_{\mathcal{P}}(A, B) := \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in B} K(\mathbf{x}, \mathbf{y})$$

is a p.d. kernel on  $\mathcal{P}(\mathcal{X})$ .

# LA kernel is p.d.: proof (6/11)

## Proof of lemma

Let  $\Phi : \mathcal{X} \mapsto \mathcal{H}$  be such that

$$K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}.$$

Then, for  $A, B \in \mathcal{P}(\mathcal{X})$ , we get:

$$\begin{aligned} K_P(A, B) &= \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in B} \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}} \\ &= \left\langle \sum_{\mathbf{x} \in A} \Phi(\mathbf{x}), \sum_{\mathbf{y} \in B} \Phi(\mathbf{y}) \right\rangle_{\mathcal{H}} \\ &= \langle \Phi_P(A), \Phi_P(B) \rangle_{\mathcal{H}}, \end{aligned}$$

with  $\Phi_P(A) := \sum_{\mathbf{x} \in A} \Phi(\mathbf{x})$ .  $\square$

## LA kernel is p.d.: proof (7/11)

### Definition: Convolution kernel (Haussler, 1999)

Let  $K_1$  and  $K_2$  be two p.d. kernels for strings. The **convolution** of  $K_1$  and  $K_2$ , denoted  $K_1 \star K_2$ , is defined for any  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$  by:

$$K_1 \star K_2(\mathbf{x}, \mathbf{y}) := \sum_{\mathbf{x}_1 \mathbf{x}_2 = \mathbf{x}, \mathbf{y}_1 \mathbf{y}_2 = \mathbf{y}} K_1(\mathbf{x}_1, \mathbf{y}_1) K_2(\mathbf{x}_2, \mathbf{y}_2).$$

### Lemma

*If  $K_1$  and  $K_2$  are p.d. then  $K_1 \star K_2$  is p.d..*

## LA kernel is p.d.: proof (8/11)

### Proof of lemma

Let  $\mathcal{X}$  be the set of finite-length strings. For  $\mathbf{x} \in \mathcal{X}$ , let

$$R(\mathbf{x}) = \{(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{X} \times \mathcal{X} : \mathbf{x} = \mathbf{x}_1\mathbf{x}_2\} \subset \mathcal{X} \times \mathcal{X}.$$

We can then write

$$K_1 \star K_2(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x}_1, \mathbf{x}_2) \in R(\mathbf{x})} \sum_{(\mathbf{y}_1, \mathbf{y}_2) \in R(\mathbf{y})} K_1(\mathbf{x}_1, \mathbf{y}_1) K_2(\mathbf{x}_2, \mathbf{y}_2)$$

which is a p.d. kernel by the previous lemmas.  $\square$

# LA kernel is p.d.: proof (9/11)

## 3 basic string kernels

- The constant kernel:

$$K_0(\mathbf{x}, \mathbf{y}) := 1.$$

- A kernel for letters:

$$K_a^{(\beta)}(\mathbf{x}, \mathbf{y}) := \begin{cases} 0 & \text{if } |\mathbf{x}| \neq 1 \text{ where } |\mathbf{y}| \neq 1, \\ \exp(\beta S(\mathbf{x}, \mathbf{y})) & \text{otherwise.} \end{cases}$$

- A kernel for gaps:

$$K_g^{(\beta)}(\mathbf{x}, \mathbf{y}) = \exp[\beta (g(|\mathbf{x}|) + g(|\mathbf{y}|))].$$

# LA kernel is p.d.: proof (10/11)

## Remark

- $S : \mathcal{A}^2 \rightarrow \mathbb{R}$  is the similarity function between letters used in the alignment score.  $K_a^{(\beta)}$  is only p.d. when the matrix:

$$(\exp(\beta s(a, b)))_{(a,b) \in \mathcal{A}^2}$$

is positive semidefinite (this is true for all  $\beta$  when  $s$  is **conditionally p.d.**).

- $g$  is the gap penalty function used in alignment score. **The gap kernel is always p.d.** (with no restriction on  $g$ ) because it can be written as:

$$K_g^{(\beta)}(\mathbf{x}, \mathbf{y}) = \exp(\beta g(|\mathbf{x}|)) \times \exp(\beta g(|\mathbf{y}|)) .$$

# LA kernel is p.d.: proof (11/11)

## Lemma

The local alignment kernel is a (limit) of convolution kernel:

$$K_{LA}^{(\beta)} = \sum_{n=0}^{\infty} K_0 \star \left( K_a^{(\beta)} \star K_g^{(\beta)} \right)^{(n-1)} \star K_a^{(\beta)} \star K_0.$$

As such it is p.d..

## Proof (sketch)

- By induction on  $n$  (simple but long to write).
- See details in Vert et al. (2004).

## LA kernel computation

- We assume an **affine gap penalty**:

$$\begin{cases} g(0) &= 0, \\ g(n) &= d + e(n - 1) \text{ si } n \geq 1, \end{cases}$$

- The LA kernel can then be computed by **dynamic programming** by:

$$K_{LA}^{(\beta)}(\mathbf{x}, \mathbf{y}) = 1 + X_2(|\mathbf{x}|, |\mathbf{y}|) + Y_2(|\mathbf{x}|, |\mathbf{y}|) + M(|\mathbf{x}|, |\mathbf{y}|),$$

where  $M(i, j)$ ,  $X(i, j)$ ,  $Y(i, j)$ ,  $X_2(i, j)$ , and  $Y_2(i, j)$  for  $0 \leq i \leq |\mathbf{x}|$ , and  $0 \leq j \leq |\mathbf{y}|$  are defined recursively.

# LA kernel is p.d.: proof (/)

## Initialization

$$\left\{ \begin{array}{l} M(i, 0) = M(0, j) = 0, \\ X(i, 0) = X(0, j) = 0, \\ Y(i, 0) = Y(0, j) = 0, \\ X_2(i, 0) = X_2(0, j) = 0, \\ Y_2(i, 0) = Y_2(0, j) = 0, \end{array} \right.$$

# LA kernel is p.d.: proof (/)

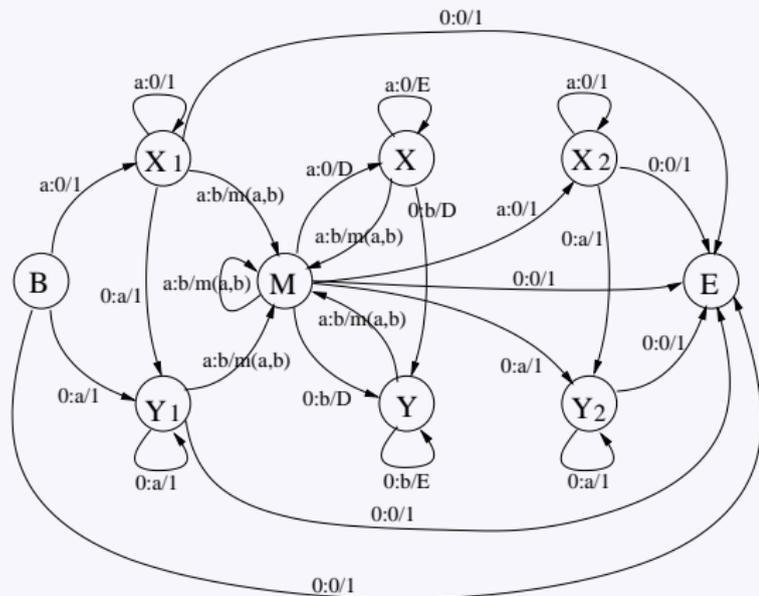
## Recursion

For  $i = 1, \dots, |\mathbf{x}|$  and  $j = 1, \dots, |\mathbf{y}|$ :

$$\begin{cases} M(i, j) &= \exp(\beta S(x_i, y_j)) \left[ 1 + X(i-1, j-1) \right. \\ &\quad \left. + Y(i-1, j-1) + M(i-1, j-1) \right], \\ X(i, j) &= \exp(\beta d) M(i-1, j) + \exp(\beta e) X(i-1, j), \\ Y(i, j) &= \exp(\beta d) [M(i, j-1) + X(i, j-1)] \\ &\quad + \exp(\beta e) Y(i, j-1), \\ X_2(i, j) &= M(i-1, j) + X_2(i-1, j), \\ Y_2(i, j) &= M(i, j-1) + X_2(i, j-1) + Y_2(i, j-1). \end{cases}$$

## LA kernel in practice

- Implementation by a finite-state transducer in  $O(|x| \times |x'|)$

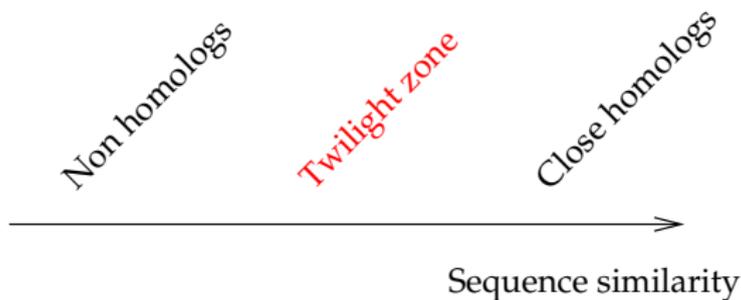


- In practice, **values are too large** (exponential scale) so taking its logarithm is a safer choice (but not p.d. anymore!)

# Outline

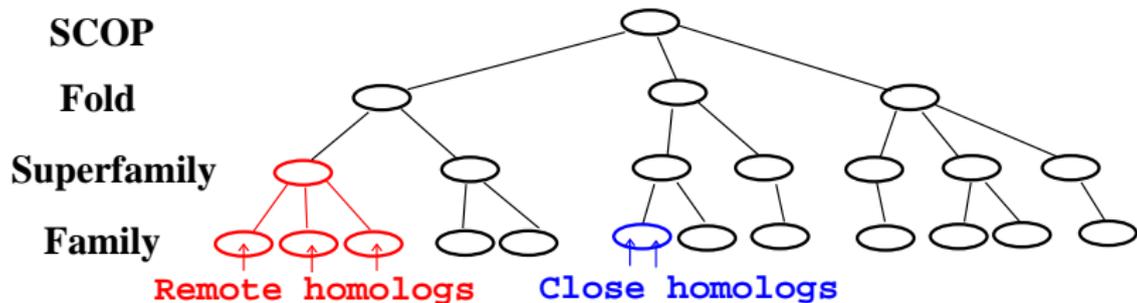
- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - **Kernels for biological sequences**
    - Motivations and history of genomics
    - Kernels derived from large feature spaces
    - Kernels derived from generative models
    - Kernels derived from a similarity measure
    - **Application to remote homology detection**
  - Kernels for graphs
  - Kernels on graphs

# Remote homology



- Homologs have **common ancestors**
- Structures and functions are more conserved than sequences
- **Remote homologs** can not be detected by direct sequence comparison

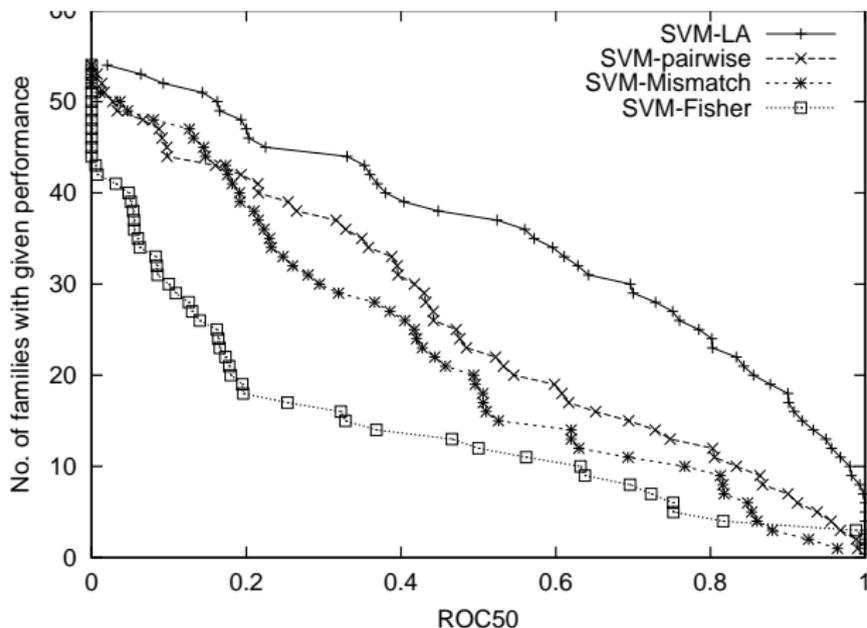
# SCOP database



## A benchmark experiment

- **Goal:** recognize directly the superfamily
- **Training:** for a sequence of interest, positive examples come from the same superfamily, but different families. Negative from other superfamilies.
- **Test:** predict the superfamily.

# Difference in performance



Performance on the SCOP superfamily recognition benchmark (from Saigo et al., 2004).

## String kernels: Summary

- A variety of principles for string kernel design have been proposed.
- Good **kernel design** is **important** for each data and each task. Performance is not the only criterion.
- Still an **art**, although principled ways have started to emerge.
- **Fast implementation** with string algorithms is often possible.
- Their application goes well beyond computational biology.

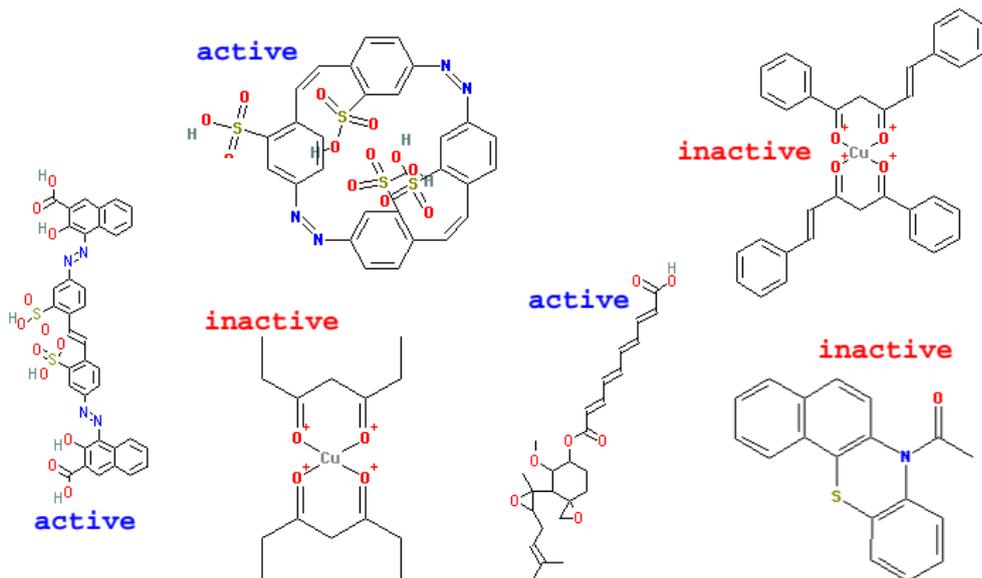
# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - **Kernels for graphs**
  - Kernels on graphs

# Outline

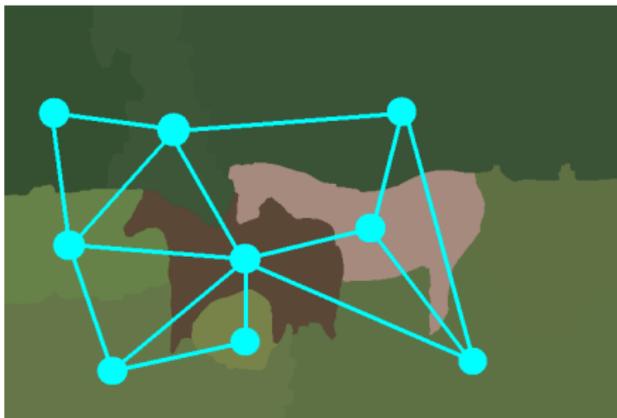
- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
    - Motivation in biology
      - Explicit enumeration of features
      - Graph kernels: challenges
      - Walk-based kernels
      - Applications
    - Kernels on graphs

# Virtual screening for drug discovery



NCI AIDS screen results (from <http://cactus.nci.nih.gov>).

# Image retrieval and classification



*From Harchaoui and Bach (2007).*

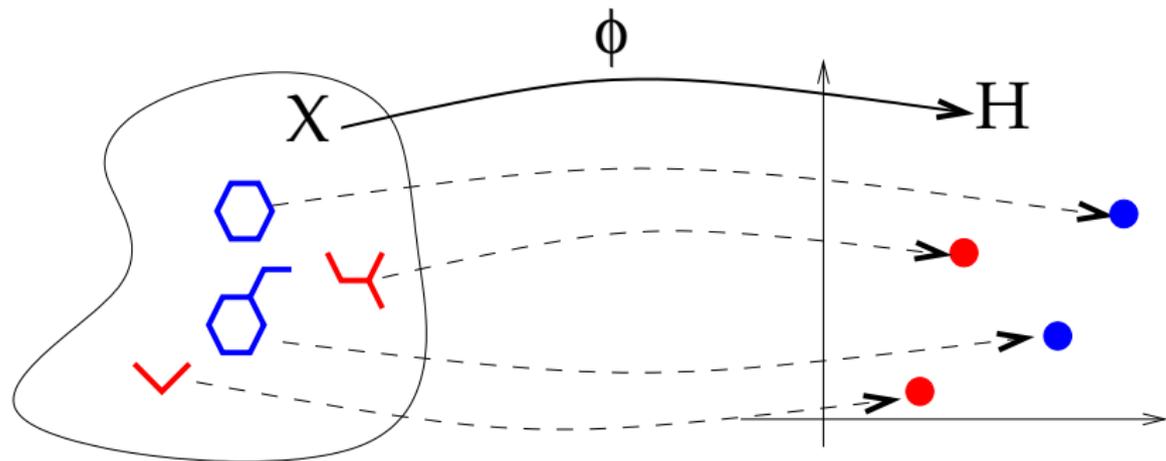
# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - **Kernels for graphs**
    - Motivation in biology
    - **Explicit enumeration of features**
    - Graph kernels: challenges
    - Walk-based kernels
    - Applications
  - Kernels on graphs

# Approach through feature enumeration

- 1 Represent each graph  $\mathbf{x}$  in  $\mathcal{X}$  by a vector  $\Phi(\mathbf{x}) \in \mathcal{H}$ , either **explicitly** or **implicitly** through the kernel

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}').$$

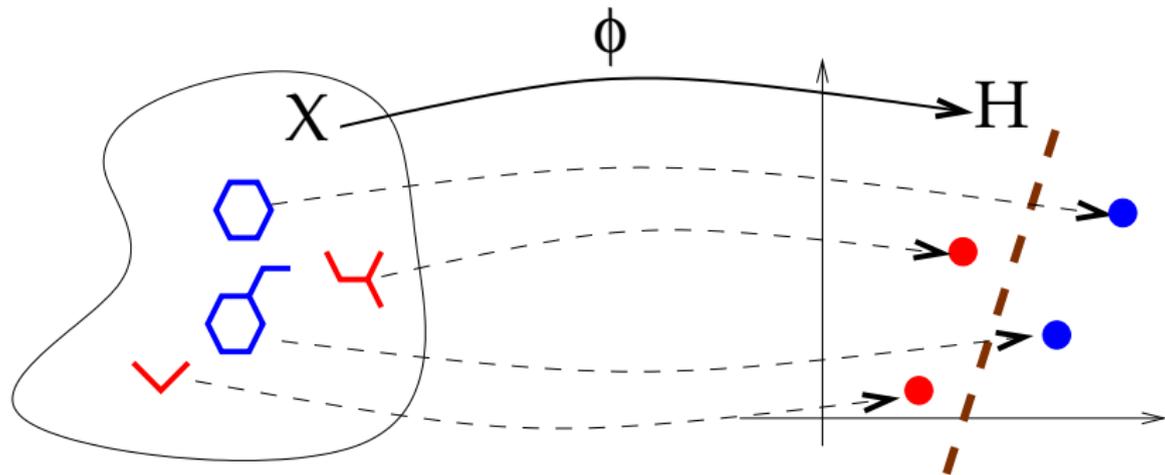


# Approach through feature enumeration

- 1 Represent each graph  $\mathbf{x}$  in  $\mathcal{X}$  by a vector  $\Phi(\mathbf{x}) \in \mathcal{H}$ , either **explicitly** or **implicitly** through the kernel

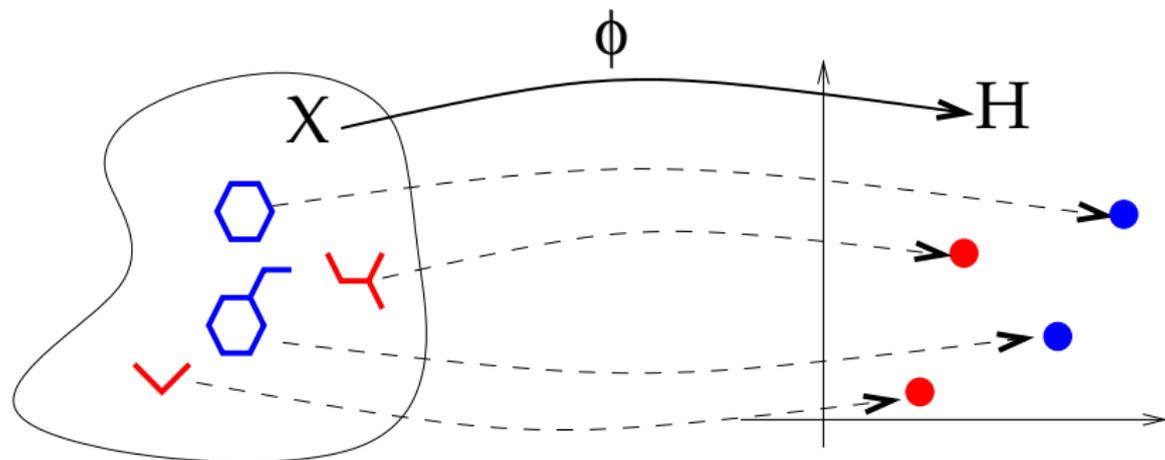
$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}').$$

- 2 Use a linear method for classification in  $\mathcal{H}$ .



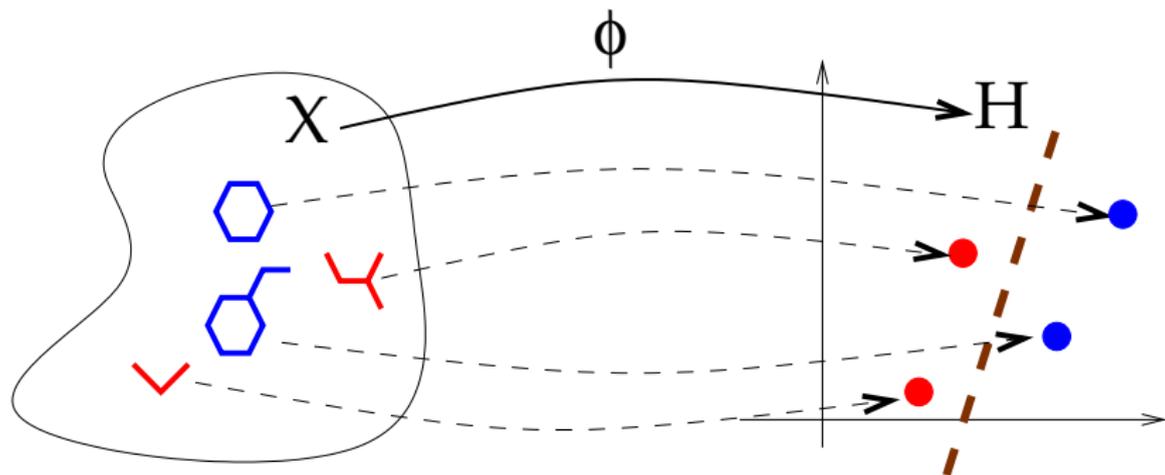
# The approach

- 1 Represent explicitly each graph  $\mathbf{x}$  by a **vector of fixed dimension**  $\Phi(\mathbf{x}) \in \mathbb{R}^p$ .



# The approach

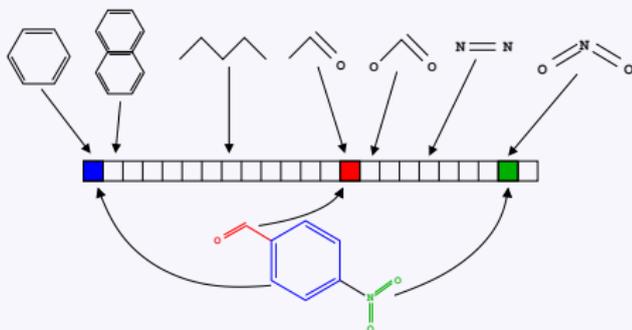
- 1 Represent explicitly each graph  $\mathbf{x}$  by a **vector of fixed dimension**  $\Phi(\mathbf{x}) \in \mathbb{R}^p$ .
- 2 Use an algorithm for **regression or pattern recognition** in  $\mathbb{R}^p$ .



# Example

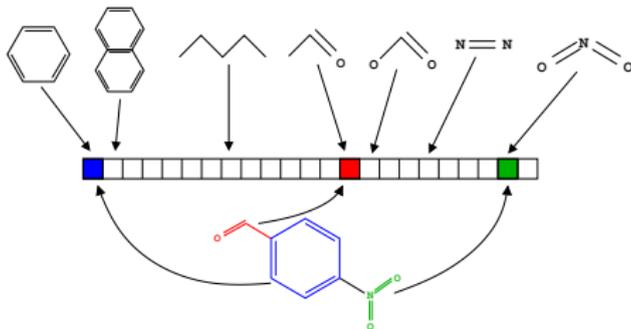
## 2D structural keys in chemoinformatics

- Index a molecule by a binary fingerprint defined by a limited set of **pre-defined** structures



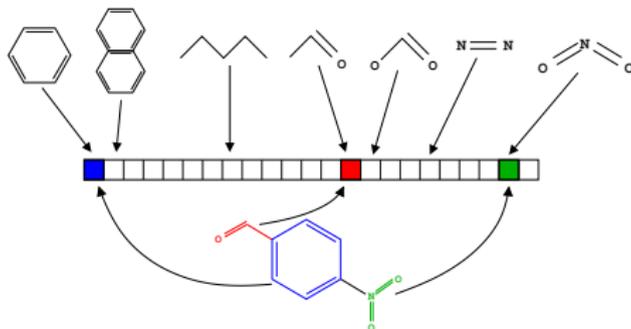
- Use a machine learning algorithms such as SVM, NN, PLS, decision tree, ...

## Challenge: which descriptors (patterns)?



- **Expressiveness**: they should retain as much information as possible from the graph
- **Computation** : they should be fast to compute
- **Large dimension** of the vector representation: memory storage, speed, statistical issues

# Indexing by substructures

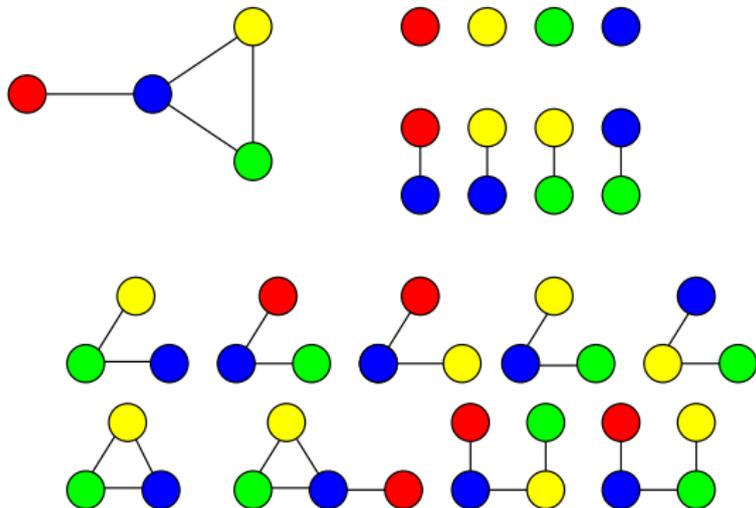


- Often we believe that **the presence substructures** are important predictive patterns
- Hence it makes sense to represent a graph by **features** that indicate the presence (or the number of occurrences) of particular substructures
- However, detecting the presence of particular substructures may be **computationally challenging**...

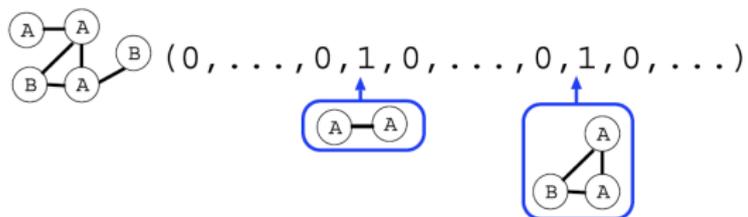
# Subgraphs

## Definition

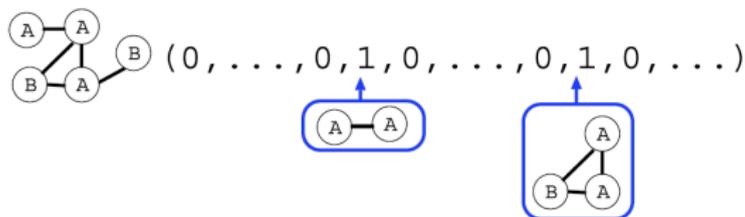
A **subgraph** of a graph  $(V, E)$  is a connected graph  $(V', E')$  with  $V' \subset V$  and  $E' \subset E$ .



# Indexing by all subgraphs?



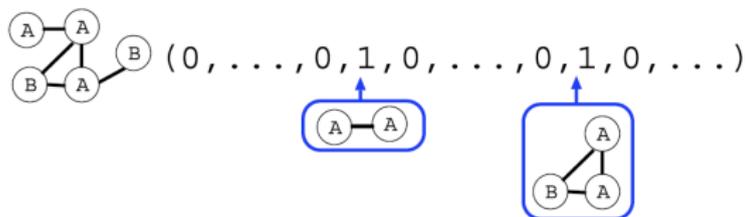
# Indexing by all subgraphs?



## Theorem

Computing all subgraph occurrences is **NP-hard**.

# Indexing by all subgraphs?



## Theorem

Computing all subgraph occurrences is **NP-hard**.

## Proof.

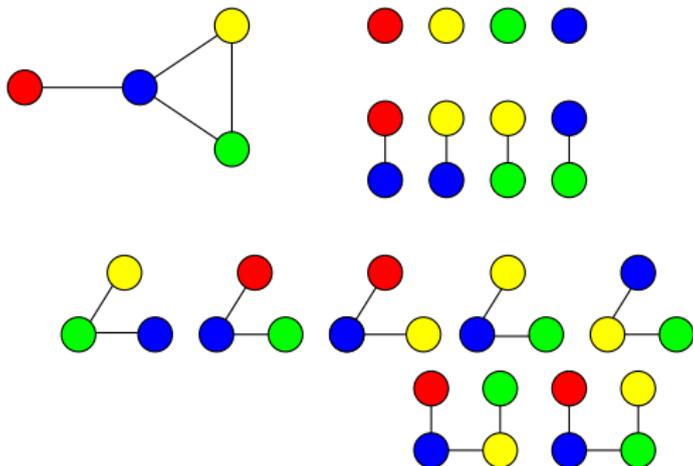
- The linear graph of size  $n$  is a subgraph of a graph  $X$  with  $n$  vertices iff  $X$  has an Hamiltonian path
- The decision problem whether a graph has a Hamiltonian path is NP-complete.



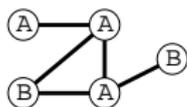
# Paths

## Definition

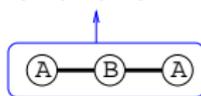
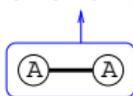
- A **path** of a graph  $(V, E)$  is sequence of **distinct vertices**  $v_1, \dots, v_n \in V$  ( $i \neq j \implies v_i \neq v_j$ ) such that  $(v_i, v_{i+1}) \in E$  for  $i = 1, \dots, n - 1$ .
- Equivalently the paths are the **linear subgraphs**.



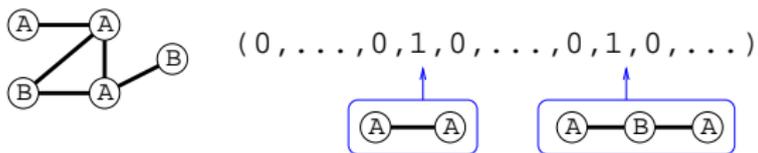
## Indexing by all paths?



$(0, \dots, 0, 1, 0, \dots, 0, 1, 0, \dots)$



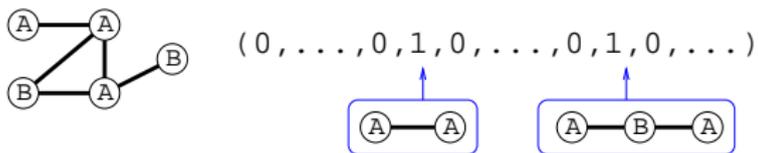
# Indexing by all paths?



## Theorem

Computing all path occurrences is *NP-hard*.

# Indexing by all paths?



## Theorem

Computing all path occurrences is *NP-hard*.

## Proof.

Same as for subgraphs. □

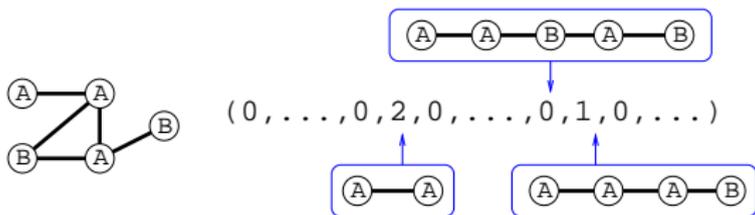
# Indexing by what?

## Substructure selection

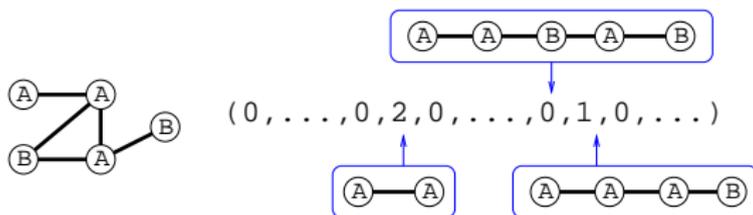
We can imagine more limited sets of substructures that lead to more computationnally efficient indexing (non-exhaustive list)

- substructures selected by **domain knowledge** (MDL fingerprint)
- all path **up to length  $k$**  (Openeye fingerprint, Nicholls 2005)
- all **shortest paths** (Borgwardt and Kriegel, 2005)
- all subgraphs **up to  $k$  vertices** (graphlet kernel, Sherashidze et al., 2009)
- all **frequent** subgraphs in the database (Helma et al., 2004)

## Example : Indexing by all shortest paths



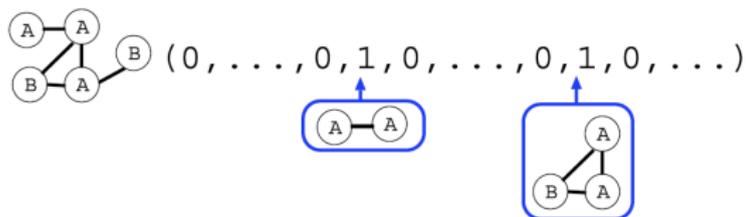
## Example : Indexing by all shortest paths



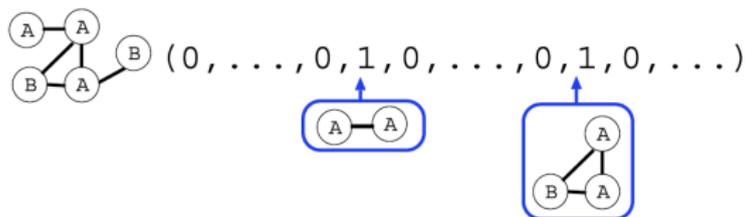
### Properties (Borgwardt and Kriegel, 2005)

- There are  $O(n^2)$  shortest paths.
- The vector of counts can be computed in  $O(n^4)$  with the Floyd-Warshall algorithm.

## Example : Indexing by all subgraphs up to $k$ vertices



## Example : Indexing by all subgraphs up to $k$ vertices



### Properties (Shervashidze et al., 2009)

- Naive enumeration scales as  $O(n^k)$ .
- Enumeration of connected graphlets in  $O(nd^{k-1})$  for graphs with degree  $\leq d$  and  $k \leq 5$ .
- Randomly sample subgraphs if enumeration is infeasible.

# Summary

- Explicit computation of substructure occurrences can be **computationally prohibitive** (subgraph, paths)
- Several ideas to **reduce** the set of substructures considered
- In practice, NP-hardness may not be so prohibitive (e.g., graphs with small degrees), the strategy followed should depend on the data considered.

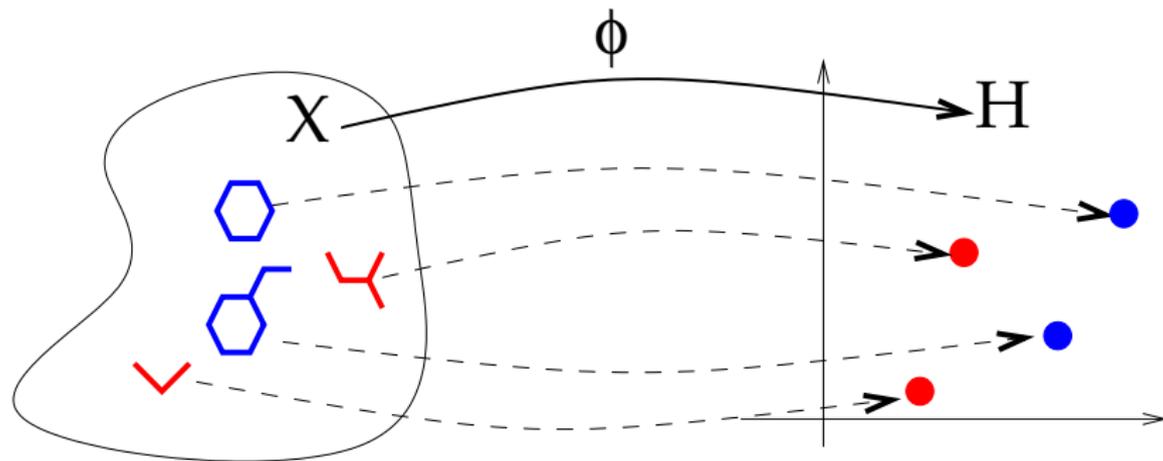
# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - **Kernels for graphs**
    - Motivation in biology
    - Explicit enumeration of features
    - **Graph kernels: challenges**
    - Walk-based kernels
    - Applications
  - Kernels on graphs

# The idea

- 1 Represent **implicitly** each graph  $\mathbf{x}$  in  $\mathcal{X}$  by a vector  $\Phi(\mathbf{x}) \in \mathcal{H}$  through the kernel

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}').$$

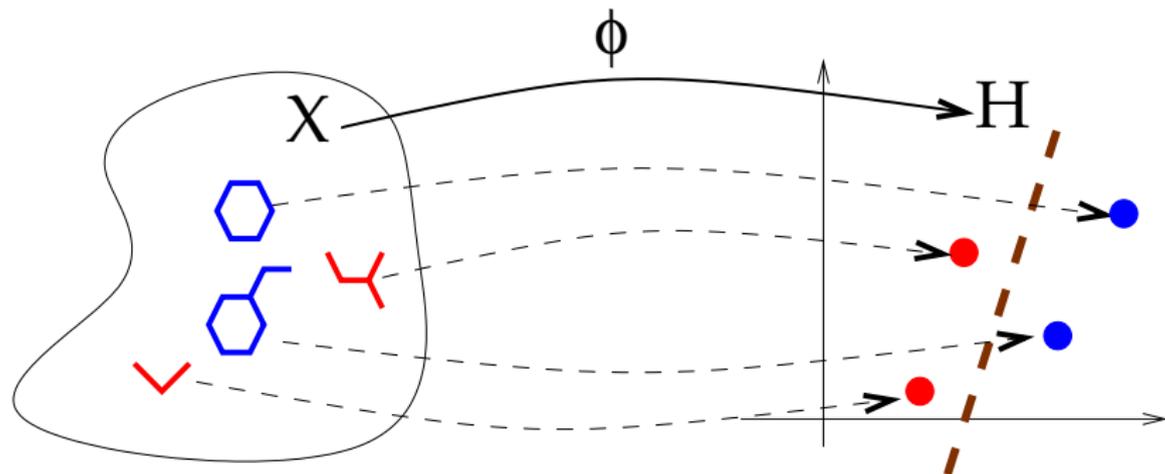


# The idea

- 1 Represent **implicitly** each graph  $\mathbf{x}$  in  $\mathcal{X}$  by a vector  $\Phi(\mathbf{x}) \in \mathcal{H}$  through the kernel

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}').$$

- 2 Use a kernel method for classification in  $\mathcal{H}$ .



# Expressiveness vs Complexity

## Definition: Complete graph kernels

A graph kernel is **complete** if it separates non-isomorphic graphs, i.e.:

$$\forall G_1, G_2 \in \mathcal{X}, \quad d_K(G_1, G_2) = 0 \implies G_1 \simeq G_2.$$

Equivalently,  $\Phi(G_1) \neq \Phi(G_2)$  if  $G_1$  and  $G_2$  are not isomorphic.

# Expressiveness vs Complexity

## Definition: Complete graph kernels

A graph kernel is **complete** if it separates non-isomorphic graphs, i.e.:

$$\forall G_1, G_2 \in \mathcal{X}, \quad d_K(G_1, G_2) = 0 \implies G_1 \simeq G_2.$$

Equivalently,  $\Phi(G_1) \neq \Phi(G_2)$  if  $G_1$  and  $G_2$  are not isomorphic.

## Expressiveness vs Complexity trade-off

- If a graph kernel is not complete, then there is **no hope** to learn all possible functions over  $\mathcal{X}$ : the kernel is not **expressive** enough.
- On the other hand, kernel **computation** must be **tractable**, i.e., no more than polynomial (with small degree) for practical applications.
- Can we define **tractable** and **expressive** graph kernels?

# Complexity of complete kernels

Proposition (Gärtner et al., 2003)

Computing **any complete graph kernel** is **at least as hard** as the graph isomorphism problem.

# Complexity of complete kernels

## Proposition (Gärtner et al., 2003)

Computing **any complete graph kernel** is **at least as hard** as the graph isomorphism problem.

## Proof

- For any kernel  $K$  the complexity of computing  $d_K$  is the same as the complexity of computing  $K$ , because:

$$d_K(G_1, G_2)^2 = K(G_1, G_1) + K(G_2, G_2) - 2K(G_1, G_2).$$

- If  $K$  is a complete graph kernel, then computing  $d_K$  solves the graph isomorphism problem ( $d_K(G_1, G_2) = 0$  iff  $G_1 \simeq G_2$ ).  $\square$

# Subgraph kernel

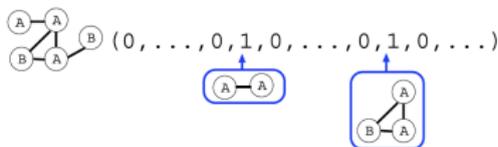
## Definition

- Let  $(\lambda_G)_{G \in \mathcal{X}}$  a set of **nonnegative** real-valued weights
- For any graph  $G \in \mathcal{X}$ , let

$$\forall H \in \mathcal{X}, \quad \Phi_H(G) = |\{G' \text{ is a subgraph of } G : G' \simeq H\}|.$$

- The **subgraph kernel** between any two graphs  $G_1$  and  $G_2 \in \mathcal{X}$  is defined by:

$$K_{\text{subgraph}}(G_1, G_2) = \sum_{H \in \mathcal{X}} \lambda_H \Phi_H(G_1) \Phi_H(G_2).$$



# Subgraph kernel complexity

Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is **NP-hard**.

# Subgraph kernel complexity

Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is **NP-hard**.

Proof (1/2)

- Let  $P_n$  be the path graph with  $n$  vertices.
- Subgraphs of  $P_n$  are path graphs:

$$\Phi(P_n) = ne_{P_1} + (n-1)e_{P_2} + \dots + e_{P_n}.$$

- The vectors  $\Phi(P_1), \dots, \Phi(P_n)$  are linearly independent, therefore:

$$e_{P_n} = \sum_{i=1}^n \alpha_i \Phi(P_i),$$

where the coefficients  $\alpha_i$  can be found in polynomial time (solving a  $n \times n$  triangular system).

# Subgraph kernel complexity

Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is **NP-hard**.

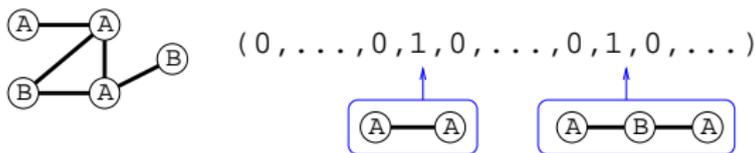
Proof (2/2)

- If  $G$  is a graph with  $n$  vertices, then it has a path that visits each node exactly once (Hamiltonian path) if and only if  $\Phi(G)^\top e_n > 0$ , i.e.,

$$\Phi(G)^\top \left( \sum_{i=1}^n \alpha_i \Phi(P_i) \right) = \sum_{i=1}^n \alpha_i K_{subgraph}(G, P_i) > 0.$$

- The decision problem whether a graph has a Hamiltonian path is NP-complete.  $\square$

# Path kernel



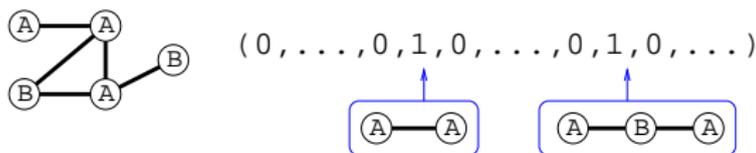
## Definition

The **path kernel** is the subgraph kernel restricted to paths, i.e.,

$$K_{path}(G_1, G_2) = \sum_{H \in \mathcal{P}} \lambda_H \Phi_H(G_1) \Phi_H(G_2),$$

where  $\mathcal{P} \subset \mathcal{X}$  is the set of path graphs.

# Path kernel



## Definition

The **path kernel** is the subgraph kernel restricted to paths, i.e.,

$$K_{path}(G_1, G_2) = \sum_{H \in \mathcal{P}} \lambda_H \Phi_H(G_1) \Phi_H(G_2),$$

where  $\mathcal{P} \subset \mathcal{X}$  is the set of path graphs.

## Proposition (Gärtner et al., 2003)

Computing the path kernel is **NP-hard**.

# Summary

## Expressiveness vs Complexity trade-off

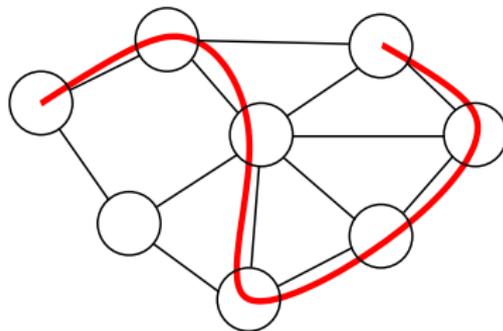
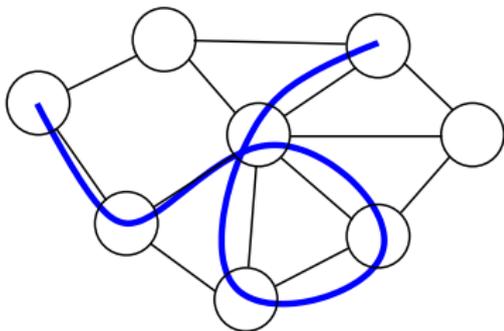
- It is **intractable** to compute **complete** graph kernels.
- It is **intractable** to compute the **subgraph kernels**.
- Restricting subgraphs to be linear does not help: it is also **intractable** to compute the **path kernel**.
- One approach to define polynomial time computable graph kernels is to have the feature space be made up of graphs **homomorphic** to subgraphs, e.g., to consider **walks** instead of paths.

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - **Kernels for graphs**
    - Motivation in biology
    - Explicit enumeration of features
    - Graph kernels: challenges
    - **Walk-based kernels**
    - Applications
  - Kernels on graphs



# Walks $\neq$ paths



# Walk kernel

## Definition

- Let  $\mathcal{S}_n$  denote the set of all possible **label sequences** of walks of length  $n$  (including vertices and edges labels), and  $\mathcal{S} = \cup_{n \geq 1} \mathcal{S}_n$ .
- For any graph  $\mathcal{X}$  let a **weight**  $\lambda_G(w)$  be associated to each walk  $w \in \mathcal{W}(G)$ .
- Let the feature vector  $\Phi(G) = (\Phi_s(G))_{s \in \mathcal{S}}$  be defined by:

$$\Phi_s(G) = \sum_{w \in \mathcal{W}(G)} \lambda_G(w) \mathbf{1}(s \text{ is the label sequence of } w) .$$

# Walk kernel

## Definition

- Let  $\mathcal{S}_n$  denote the set of all possible **label sequences** of walks of length  $n$  (including vertices and edges labels), and  $\mathcal{S} = \cup_{n \geq 1} \mathcal{S}_n$ .
- For any graph  $\mathcal{X}$  let a **weight**  $\lambda_G(w)$  be associated to each walk  $w \in \mathcal{W}(G)$ .
- Let the feature vector  $\Phi(G) = (\Phi_s(G))_{s \in \mathcal{S}}$  be defined by:

$$\Phi_s(G) = \sum_{w \in \mathcal{W}(G)} \lambda_G(w) \mathbf{1}(s \text{ is the label sequence of } w) .$$

- A walk kernel is a graph kernel defined by:

$$K_{walk}(G_1, G_2) = \sum_{s \in \mathcal{S}} \Phi_s(G_1) \Phi_s(G_2) .$$

# Walk kernel examples

## Examples

- The  $n$ th-order walk kernel is the walk kernel with  $\lambda_G(w) = 1$  if the length of  $w$  is  $n$ , 0 otherwise. It compares two graphs through their common walks of length  $n$ .

# Walk kernel examples

## Examples

- The  *$n$ th-order walk kernel* is the walk kernel with  $\lambda_G(w) = 1$  if the length of  $w$  is  $n$ , 0 otherwise. It compares two graphs through their common walks of length  $n$ .
- The *random walk kernel* is obtained with  $\lambda_G(w) = P_G(w)$ , where  $P_G$  is a *Markov random walk on  $G$* . In that case we have:

$$K(G_1, G_2) = P(\text{label}(W_1) = \text{label}(W_2)),$$

where  $W_1$  and  $W_2$  are two independent random walks on  $G_1$  and  $G_2$ , respectively (Kashima et al., 2003).

# Walk kernel examples

## Examples

- The  *$n$ th-order walk kernel* is the walk kernel with  $\lambda_G(w) = 1$  if the length of  $w$  is  $n$ , 0 otherwise. It compares two graphs through their common walks of length  $n$ .
- The *random walk kernel* is obtained with  $\lambda_G(w) = P_G(w)$ , where  $P_G$  is a *Markov random walk on  $G$* . In that case we have:

$$K(G_1, G_2) = P(\text{label}(W_1) = \text{label}(W_2)),$$

where  $W_1$  and  $W_2$  are two independent random walks on  $G_1$  and  $G_2$ , respectively (Kashima et al., 2003).

- The *geometric walk kernel* is obtained (when it converges) with  $\lambda_G(w) = \beta^{\text{length}(w)}$ , for  $\beta > 0$ . In that case the feature space is of *infinite dimension* (Gärtner et al., 2003).

# Computation of walk kernels

## Proposition

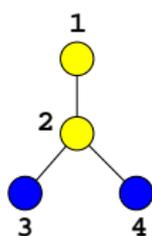
These three kernels ( $n$ th-order, random and geometric walk kernels) can be computed efficiently in **polynomial time**.

# Product graph

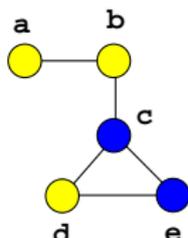
## Definition

Let  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  be two graphs with labeled vertices. The **product graph**  $G = G_1 \times G_2$  is the graph  $G = (V, E)$  with:

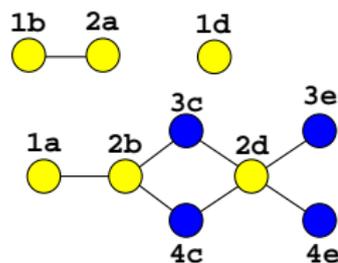
- 1  $V = \{(v_1, v_2) \in V_1 \times V_2 : v_1 \text{ and } v_2 \text{ have the same label}\}$ ,
- 2  $E = \{((v_1, v_2), (v'_1, v'_2)) \in V \times V : (v_1, v'_1) \in E_1 \text{ and } (v_2, v'_2) \in E_2\}$ .



**G1**



**G2**



**G1 x G2**

# Walk kernel and product graph

## Lemma

There is a **bijection** between:

- 1 The **pairs of walks**  $w_1 \in \mathcal{W}_n(G_1)$  and  $w_2 \in \mathcal{W}_n(G_2)$  with the **same label sequences**,
- 2 The **walks on the product graph**  $w \in \mathcal{W}_n(G_1 \times G_2)$ .

# Walk kernel and product graph

## Lemma

There is a **bijection** between:

- 1 The **pairs of walks**  $w_1 \in \mathcal{W}_n(G_1)$  and  $w_2 \in \mathcal{W}_n(G_2)$  with the **same label sequences**,
- 2 The **walks on the product graph**  $w \in \mathcal{W}_n(G_1 \times G_2)$ .

## Corollary

$$\begin{aligned} K_{\text{walk}}(G_1, G_2) &= \sum_{s \in \mathcal{S}} \Phi_s(G_1) \Phi_s(G_2) \\ &= \sum_{(w_1, w_2) \in \mathcal{W}(G_1) \times \mathcal{W}(G_2)} \lambda_{G_1}(w_1) \lambda_{G_2}(w_2) \mathbf{1}(l(w_1) = l(w_2)) \\ &= \sum_{w \in \mathcal{W}(G_1 \times G_2)} \lambda_{G_1 \times G_2}(w). \end{aligned}$$

## Computation of the $n$ th-order walk kernel

- For the  $n$ th-order walk kernel we have  $\lambda_{G_1 \times G_2}(w) = 1$  if the length of  $w$  is  $n$ , 0 otherwise.
- Therefore:

$$K_{nth-order}(G_1, G_2) = \sum_{w \in \mathcal{W}_n(G_1 \times G_2)} 1.$$

- Let  $A$  be the adjacency matrix of  $G_1 \times G_2$ . Then we get:

$$K_{nth-order}(G_1, G_2) = \sum_{i,j} [A^n]_{i,j} = \mathbf{1}^\top A^n \mathbf{1}.$$

- Computation in  $O(n|G_1||G_2|d_1d_2)$ , where  $d_i$  is the maximum degree of  $G_i$ .

# Computation of random and geometric walk kernels

- In both cases  $\lambda_G(w)$  for a walk  $w = v_1 \dots v_n$  can be decomposed as:

$$\lambda_G(v_1 \dots v_n) = \lambda^i(v_1) \prod_{i=2}^n \lambda^t(v_{i-1}, v_i).$$

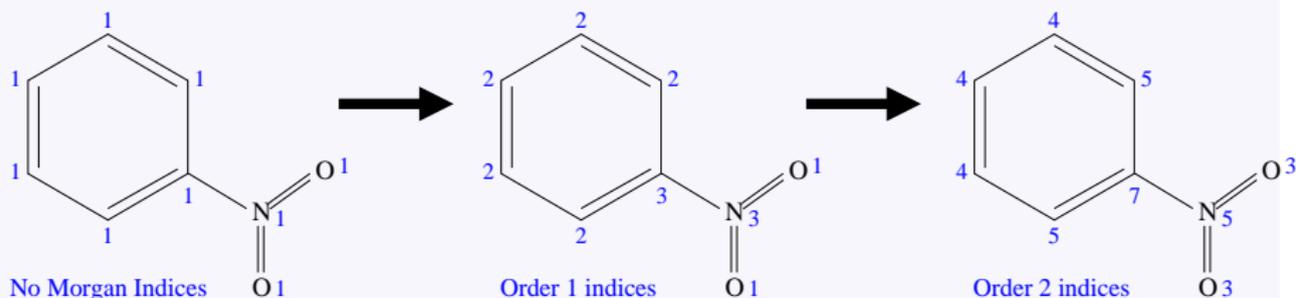
- Let  $\Lambda_i$  be the vector of  $\lambda^i(v)$  and  $\Lambda_t$  be the matrix of  $\lambda^t(v, v')$ :

$$\begin{aligned} K_{walk}(G_1, G_2) &= \sum_{n=1}^{\infty} \sum_{w \in \mathcal{W}_n(G_1 \times G_2)} \lambda^i(v_1) \prod_{i=2}^n \lambda^t(v_{i-1}, v_i) \\ &= \sum_{n=0}^{\infty} \Lambda_i \Lambda_t^n \mathbf{1} \\ &= \Lambda_i (I - \Lambda_t)^{-1} \mathbf{1} \end{aligned}$$

- Computation in  $O(|G_1|^3 |G_2|^3)$

# Extensions 1: label enrichment

## Atom relabeling with the Morgan index (Mahé et al., 2004)

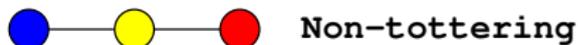


- **Compromise** between **fingerprints** and **structural keys features**.
- Other **relabeling** schemes are possible (graph coloring).
- **Faster computation with more labels** (less matches implies a smaller product graph).

## Extension 2: Non-tottering walk kernel

### Tottering walks

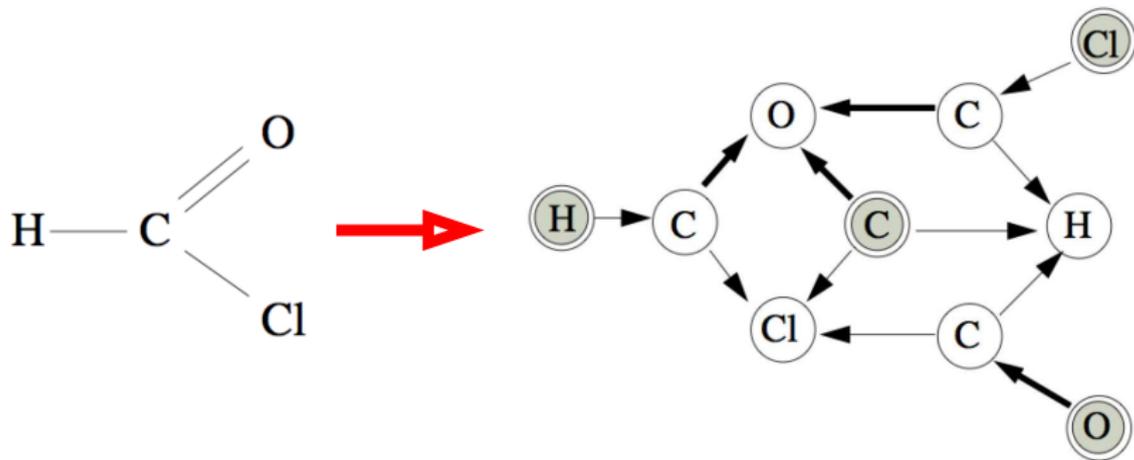
A **tottering walk** is a walk  $w = v_1 \dots v_n$  with  $v_i = v_{i+2}$  for some  $i$ .



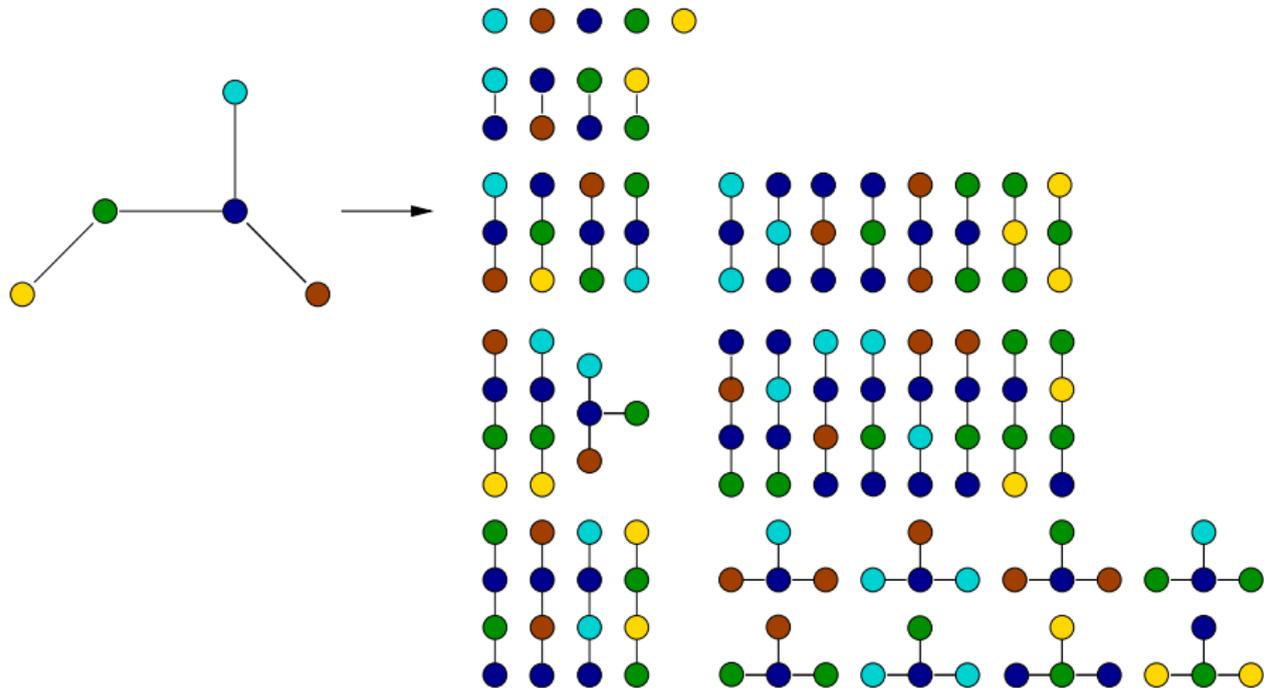
- Tottering walks seem **irrelevant** for many applications
- Focusing on non-tottering walks is a way to get closer to the **path kernel** (e.g., equivalent on trees).

# Computation of the non-tottering walk kernel (Mahé et al., 2005)

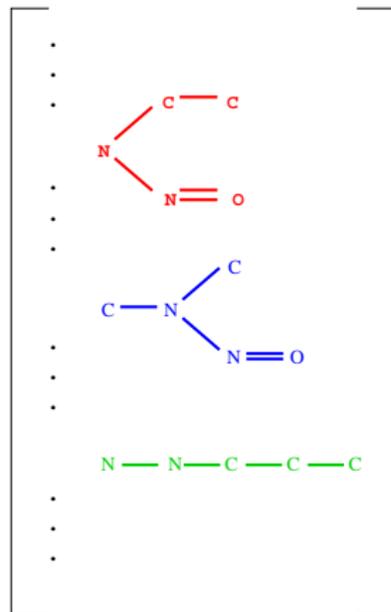
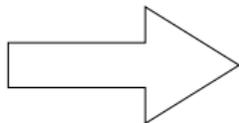
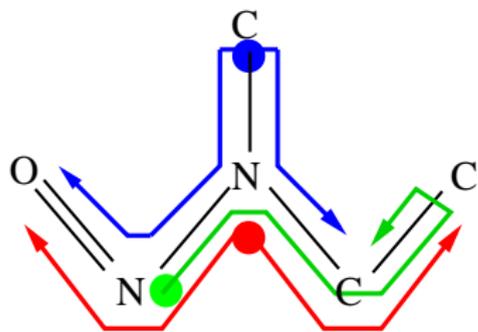
- **Second-order** Markov random walk to prevent tottering walks
- Written as a **first-order** Markov random walk on an **augmented graph**
- **Normal** walk kernel on the augmented graph (which is always a **directed** graph).



## Extension 3: Subtree kernels



# Example: Tree-like fragments of molecules



# Computation of the subtree kernel (Ramon and Gärtner, 2003; Mahé and Vert, 2009)

- Like the walk kernel, amounts to compute the (weighted) number of subtrees in the **product graph**.
- Recursion: if  $\mathcal{T}(v, n)$  denotes the weighted number of subtrees of depth  $n$  rooted at the vertex  $v$ , then:

$$\mathcal{T}(v, n+1) = \sum_{R \subset \mathcal{N}(v)} \prod_{v' \in R} \lambda_t(v, v') \mathcal{T}(v', n),$$

where  $\mathcal{N}(v)$  is the set of neighbors of  $v$ .

- Can be combined with the non-tottering graph transformation as preprocessing to obtain the **non-tottering subtree kernel**.

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - **Kernels for graphs**
    - Motivation in biology
    - Explicit enumeration of features
    - Graph kernels: challenges
    - Walk-based kernels
    - **Applications**
  - Kernels on graphs

# Application in chemoinformatics (Mahé et al., 2005)

## MUTAG dataset

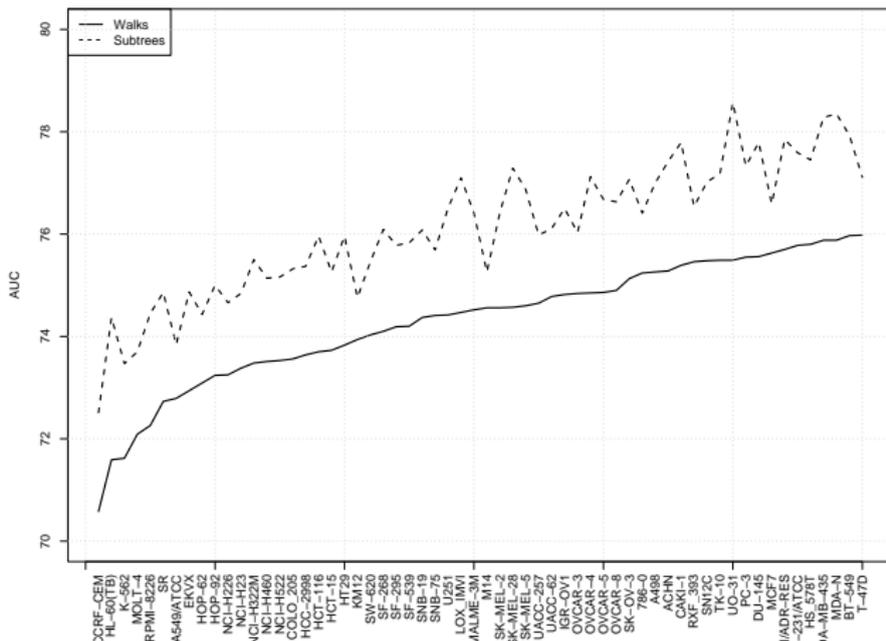
- aromatic/hetero-aromatic compounds
- high mutagenic activity / no mutagenic activity, assayed in *Salmonella typhimurium*.
- 188 compounds: 125 + / 63 -

## Results

10-fold cross-validation accuracy

Method	Accuracy
Progol1	81.4%
2D kernel	91.2%

## 2D Subtree vs walk kernels

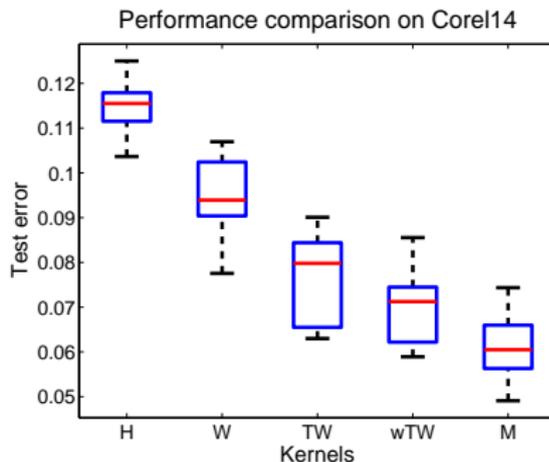


Screening of inhibitors for 60 cancer cell lines.

# Image classification (Harchaoui and Bach, 2007)

## COREL14 dataset

- 1400 natural images in 14 classes
- Compare kernel between histograms (H), walk kernel (W), subtree kernel (TW), weighted subtree kernel (wTW), and a combination (M).



# Summary: graph kernels

## What we saw

- Kernels do **not allow** to overcome the NP-hardness of subgraph patterns
- They allow to work with approximate subgraphs (walks, subtrees), in infinite dimension, thanks to the **kernel trick**
- However: using kernels makes it difficult to **come back to patterns** after the learning stage

# Outline

- 1 Kernels and RKHS
- 2 Kernels Methods: Supervised Learning
- 3 Kernels Methods: Unsupervised Learning
- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - **Kernels on graphs**

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - **Kernels on graphs**
    - Motivation
      - Graph distance and p.d. kernels
      - Construction by regularization
      - The diffusion kernel
      - Harmonic analysis on graphs
      - Applications

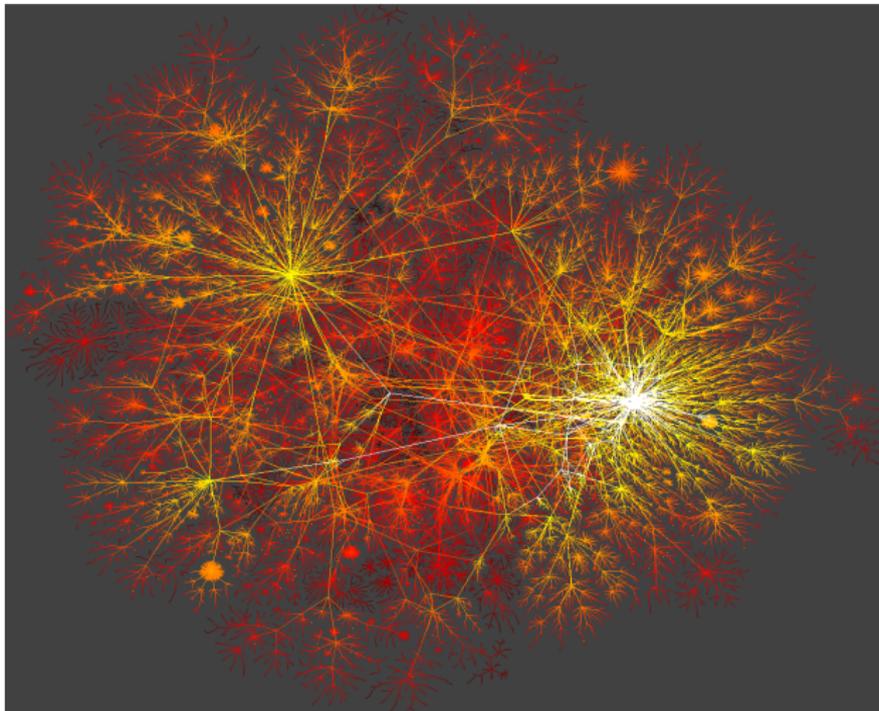
# Graphs

## Motivation

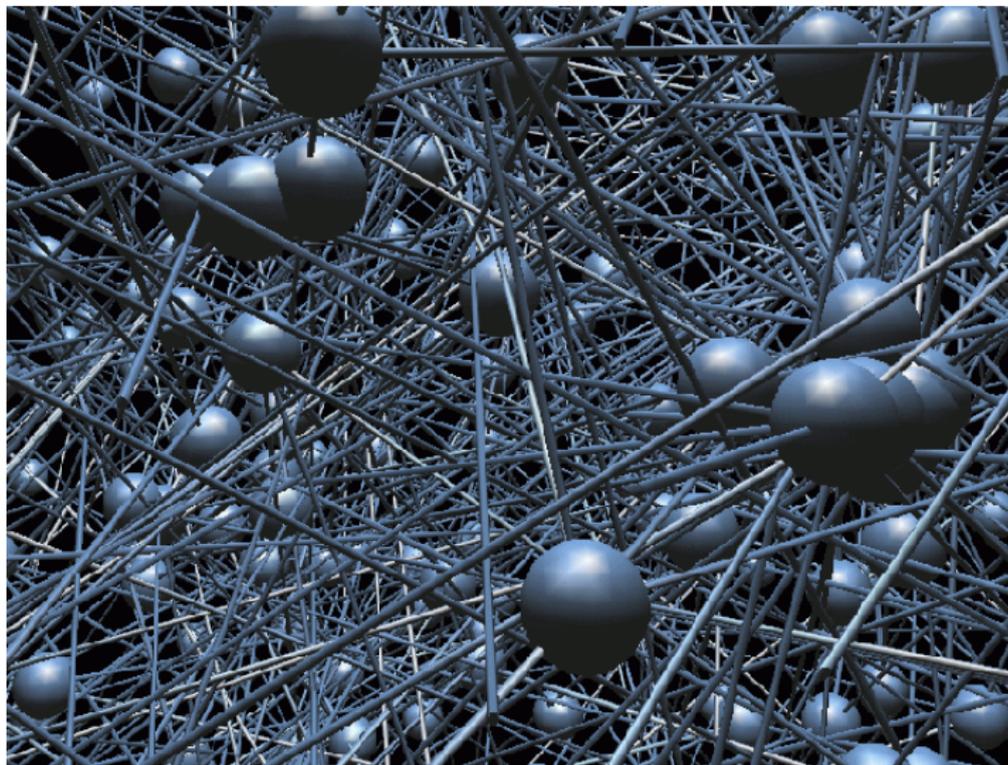
Many data come in the form of **nodes in a graph** for different reasons:

- by **definition** (interaction network, internet...)
- by **discretization** / sampling of a continuous domain
- by **convenience** (e.g., if only a similarity function is available)

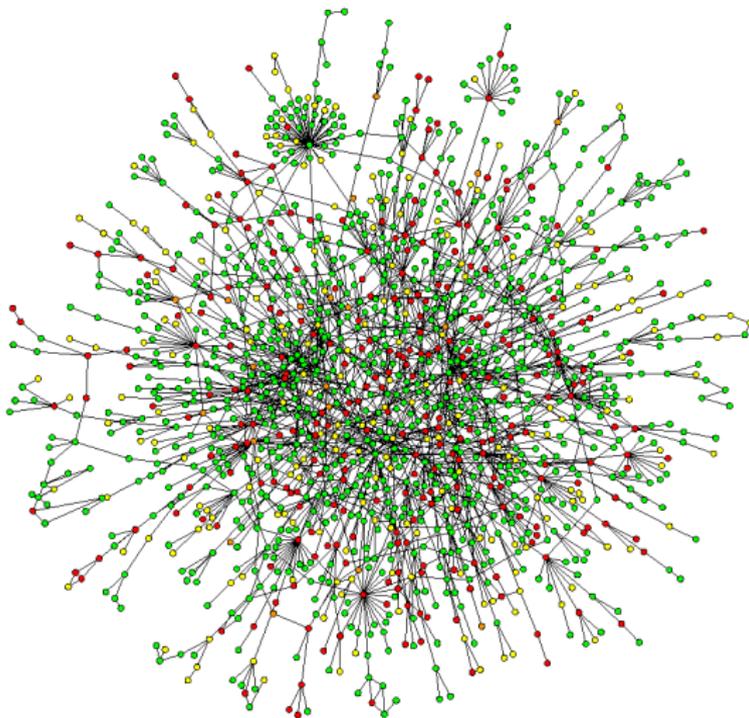
# Example: web



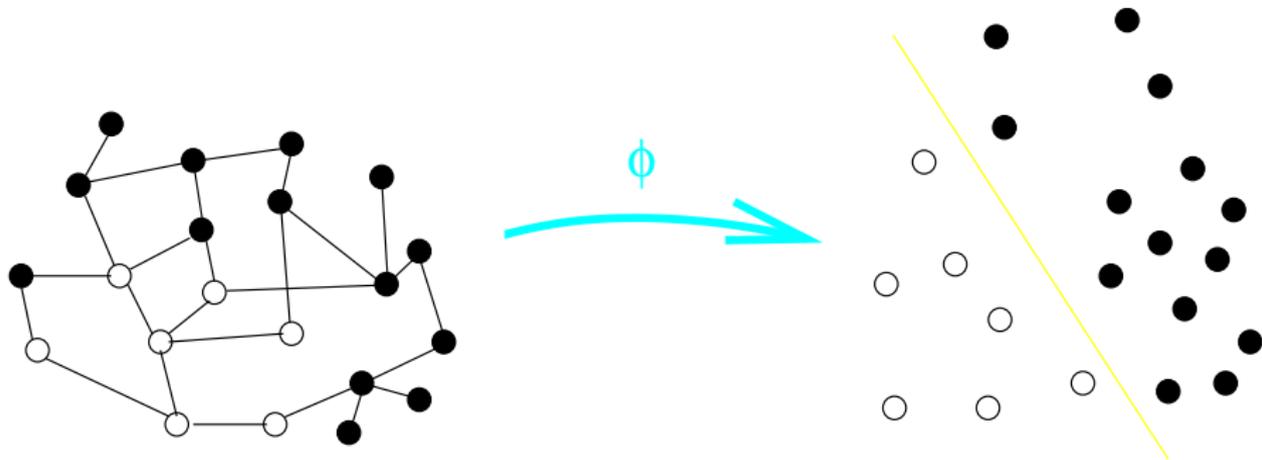
## Example: social network



## Example: protein-protein interaction



## Kernel on a graph



- We need a **kernel  $K(\mathbf{x}, \mathbf{x}')$**  between nodes of the graph.
- Example: predict gene protein functions from high-throughput protein-protein interaction data.

# General remarks

## Strategies to make a kernel on a graph

- $\mathcal{X}$  being finite, **any symmetric semi-definite matrix  $K$**  defines a valid p.d. kernel on  $\mathcal{X}$ .

# General remarks

## Strategies to make a kernel on a graph

- $\mathcal{X}$  being finite, **any symmetric semi-definite matrix  $K$**  defines a valid p.d. kernel on  $\mathcal{X}$ .
- How to “translate” the graph topology into the kernel?
  - **Direct geometric approach:**  $K_{i,j}$  should be “large” when  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are “close” to each other on the graph?
  - **Functional approach:**  $\|f\|_K$  should be “small” when  $f$  is “smooth” on the graph?
  - **Link discrete/continuous:** is there an equivalent to the continuous Gaussian kernel on the graph (e.g., limit by fine discretization)?

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - **Kernels on graphs**
    - Motivation
    - **Graph distance and p.d. kernels**
    - Construction by regularization
    - The diffusion kernel
    - Harmonic analysis on graphs
    - Applications

# Conditionally p.d. kernels

## Hilbert distance

- Any p.d. kernels is an inner product in a Hilbert space

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}} .$$

- It defines a Hilbert distance:

$$d_K(\mathbf{x}, \mathbf{x}')^2 = K(\mathbf{x}, \mathbf{x}) + K(\mathbf{x}', \mathbf{x}') - 2K(\mathbf{x}, \mathbf{x}')$$

- $-d_K^2$  is **conditionally positive definite (c.p.d.)**, i.e.:

$$\forall t > 0, \quad \exp\left(-td_K(\mathbf{x}, \mathbf{x}')^2\right) \text{ is p.d.}$$

# Example

## A direct approach

- For  $\mathcal{X} = \mathbb{R}^n$ , the inner product is p.d.:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'.$$

- The corresponding Hilbert distance is the Euclidean distance:

$$d_K(\mathbf{x}, \mathbf{x}')^2 = \mathbf{x}^\top \mathbf{x} + \mathbf{x}'^\top \mathbf{x}' - 2\mathbf{x}^\top \mathbf{x}' = \|\mathbf{x} - \mathbf{x}'\|^2.$$

- $-d_K^2$  is **conditionally positive definite (c.p.d.)**, i.e.:

$$\forall t > 0, \quad \exp(-t\|\mathbf{x} - \mathbf{x}'\|^2) \text{ is p.d.}$$

# Graph distance

## Graph embedding in a Hilbert space

- Given a graph  $G = (V, E)$ , the **graph distance**  $d_G(x, x')$  between any two vertices is the **length of the shortest path** between  $x$  and  $x'$ .
- We say that the graph  $G = (V, E)$  can be **embedded** (exactly) in a Hilbert space if  $-d_G^2$  is **c.p.d.**, which implies in particular that  $\exp(-td_G(x, x'))$  is p.d. for all  $t > 0$ .

# Graph distance

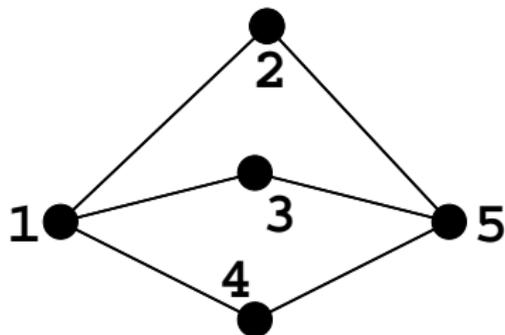
## Graph embedding in a Hilbert space

- Given a graph  $G = (V, E)$ , the **graph distance**  $d_G(x, x')$  between any two vertices is the **length of the shortest path** between  $x$  and  $x'$ .
- We say that the graph  $G = (V, E)$  can be **embedded** (exactly) in a Hilbert space if  $-d_G^2$  is **c.p.d.**, which implies in particular that  $\exp(-td_G(x, x'))$  is p.d. for all  $t > 0$ .

## Lemma

- *In general graphs can not be embedded exactly in Hilbert spaces.*
- *In some cases exact embeddings exists, e.g.:*
  - *trees can be embedded exactly,*
  - *closed chains can be embedded exactly.*

## Example: non-c.p.d. graph distance



$$d_G = \begin{pmatrix} 0 & 1 & 1 & 1 & 2 \\ 1 & 0 & 2 & 2 & 1 \\ 1 & 2 & 0 & 2 & 1 \\ 1 & 2 & 2 & 0 & 1 \\ 2 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\lambda_{\min} \left( \left[ e^{(-0.2d_G(i,j))} \right] \right) = -0.028 < 0.$$

# Graph distance on trees are c.p.d.

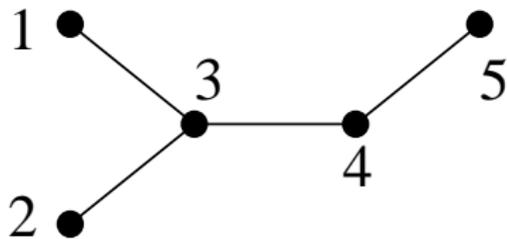
## Proof

- Let  $G = (V, E)$  a tree
- Fix a root  $x_0 \in V$
- Represent any vertex  $x \in V$  by a vector  $\Phi(x) \in \mathbb{R}^{|E|}$ , where  $\Phi(x)_i = 1$  if the  $i$ -th edge is in the (unique) path between  $x$  and  $x_0$ , 0 otherwise.
- Then:

$$d_G(x, x') = \|\Phi(x) - \Phi(x')\|^2,$$

and therefore  $-d_G$  is c.p.d., in particular  $\exp(-td_G(x, x'))$  is p.d. for all  $t > 0$ .

## Example

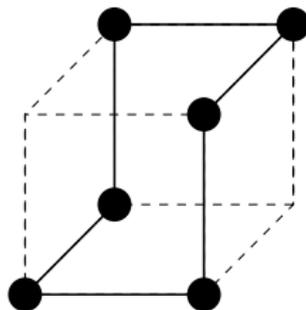
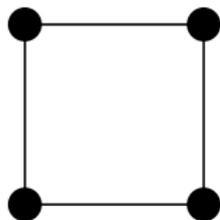


$$\left[ e^{-d_G(i,j)} \right] = \begin{pmatrix} 1 & 0.14 & 0.37 & 0.14 & 0.05 \\ 0.14 & 1 & 0.37 & 0.14 & 0.05 \\ 0.37 & 0.37 & 1 & 0.37 & 0.14 \\ 0.14 & 0.14 & 0.37 & 1 & 0.37 \\ 0.05 & 0.05 & 0.14 & 0.37 & 1 \end{pmatrix}$$

## Graph distance on closed chains are c.p.d.

Proof: case  $|V| = 2p$

- Let  $G = (V, E)$  a cycle with an even number of vertices  $|V| = 2p$
- Fix a root  $x_0 \in V$ , number the  $2p$  edges from  $x_0$  to  $x_0$ .
- Map the  $2p$  edges in  $\mathbb{R}^p$  to  $(e_1, \dots, e_p, -e_1, \dots, -e_p)$
- Map a vertex  $v$  to the sum of the edges in the shortest path between  $x_0$  and  $v$ .



# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - **Kernels on graphs**
    - Motivation
    - Graph distance and p.d. kernels
    - **Construction by regularization**
    - The diffusion kernel
    - Harmonic analysis on graphs
    - Applications

# Functional approach

## Motivation

- How to make p.d. kernel on **general graphs**?
- Making a kernel is equivalent to defining a **RKHS**.
- There are intuitive notions of **smoothness** on a graph

## Idea

- Define a priori a **smoothness functional** on the functions  $f : \mathcal{X} \rightarrow \mathbb{R}$ .
- Show that **it defines a RKHS** and identify the corresponding kernel

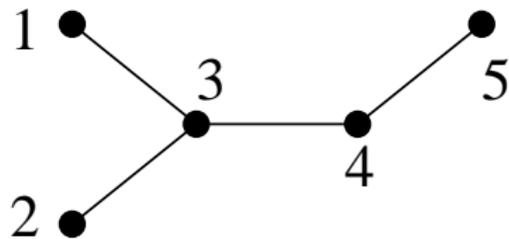
# Notations

- $\mathcal{X} = (\mathbf{x}_1, \dots, \mathbf{x}_m)$  is finite.
- For  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , we note  $\mathbf{x} \sim \mathbf{x}'$  to indicate the existence of an edge between  $\mathbf{x}$  and  $\mathbf{x}'$
- We assume that there is **no self-loop**  $\mathbf{x} \sim \mathbf{x}$ , and that there is **a single connected component**.
- The **adjacency matrix** is  $A \in \mathbb{R}^{m \times m}$ :

$$A_{i,j} = \begin{cases} 1 & \text{if } i \sim j, \\ 0 & \text{otherwise.} \end{cases}$$

- $D$  is the diagonal matrix where  $D_{i,i}$  is the number of neighbors of  $\mathbf{x}_i$  ( $D_{i,i} = \sum_{j=1}^m A_{i,j}$ ).

## Example



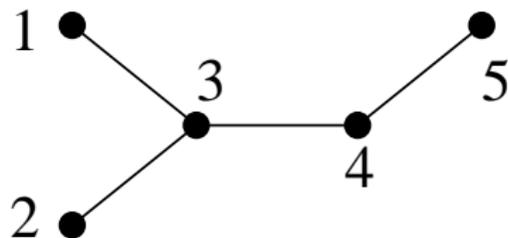
$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

# Graph Laplacian

## Definition

The Laplacian of the graph is the **matrix**  $L = D - A$ .



$$L = D - A = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

# Properties of the Laplacian

## Lemma

Let  $L = D - A$  be the Laplacian of a *connected* graph:

- For any  $f : \mathcal{X} \rightarrow \mathbb{R}$ ,

$$\Omega(f) := \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = f^\top L f$$

- $L$  is a *symmetric positive semi-definite* matrix
- 0 is an *eigenvalue* with multiplicity 1 associated to the constant eigenvector  $\mathbf{1} = (1, \dots, 1)$
- The *image* of  $L$  is

$$\text{Im}(L) = \left\{ f \in \mathbb{R}^m : \sum_{i=1}^m f_i = 0 \right\}$$

## Proof: link between $\Omega(f)$ and $L$

$$\begin{aligned}\Omega(f) &= \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 \\ &= \sum_{i \sim j} \left( f(\mathbf{x}_i)^2 + f(\mathbf{x}_j)^2 - 2f(\mathbf{x}_i)f(\mathbf{x}_j) \right) \\ &= \sum_{i=1}^m D_{i,i} f(\mathbf{x}_i)^2 - 2 \sum_{i \sim j} f(\mathbf{x}_i) f(\mathbf{x}_j) \\ &= \mathbf{f}^\top D \mathbf{f} - \mathbf{f}^\top A \mathbf{f} \\ &= \mathbf{f}^\top L \mathbf{f}\end{aligned}$$

## Proof: eigenstructure of $L$

- $L$  is symmetric because  $A$  and  $D$  are symmetric.
- For any  $f \in \mathbb{R}^m$ ,  $f^\top Lf = \Omega(f) \geq 0$ , therefore the (real-valued) eigenvalues of  $L$  are  $\geq 0$  :  $L$  is therefore positive semi-definite.
- $f$  is an eigenvector associated to eigenvalue 0  
iff  $f^\top Lf = 0$   
iff  $\sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = 0$  ,  
iff  $f(\mathbf{x}_i) = f(\mathbf{x}_j)$  when  $i \sim j$ ,  
iff  $f$  is constant (because the graph is connected).
- $L$  being symmetric,  $Im(L)$  is the orthogonal supplement of  $Ker(L)$ , that is, the set of functions orthogonal to  $\mathbf{1}$ .  $\square$

# Our first graph kernel

## Theorem

The set  $\mathcal{H} = \{f \in \mathbb{R}^m : \sum_{i=1}^m f_i = 0\}$  endowed with the norm:

$$\Omega(f) = \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2$$

is a RKHS whose reproducing kernel is  $L^*$ , the pseudo-inverse of the graph Laplacian.

## In case of...

### Pseudo-inverse of $L$

Remember the pseudo-inverse  $L^*$  of  $L$  is the linear application that is equal to:

- 0 on  $\text{Ker}(L)$
- $L^{-1}$  on  $\text{Im}(L)$ , that is, if we write:

$$L = \sum_{i=1}^m \lambda_i u_i u_i^\top$$

the eigendecomposition of  $L$ :

$$L^* = \sum_{\lambda_i \neq 0} (\lambda_i)^{-1} u_i u_i^\top.$$

- In particular it holds that  $L^*L = LL^* = \Pi_{\mathcal{H}}$ , the projection onto  $\text{Im}(L) = \mathcal{H}$ .

## Proof (1/2)

- Restricted to  $\mathcal{H}$ , the symmetric bilinear form:

$$\langle f, g \rangle = f^\top Lg$$

is positive definite (because  $L$  is positive semi-definite, and  $\mathcal{H} = \text{Im}(L)$ ). It is therefore a scalar product, making of  $\mathcal{H}$  a **Hilbert space** (in fact Euclidean).

- The norm in this Hilbert space  $\mathcal{H}$  is:

$$\|f\|^2 = \langle f, f \rangle = f^\top Lf = \Omega(f).$$

## Proof (2/2)

To check that  $\mathcal{H}$  is a RKHS with reproducing kernel  $K = L^*$ , it suffices to show that:

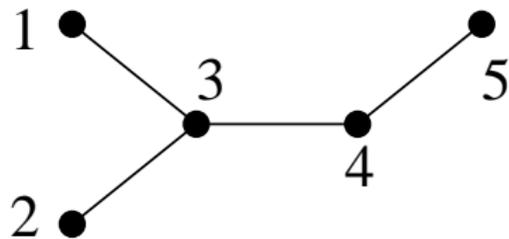
$$\begin{cases} \forall \mathbf{x} \in \mathcal{X}, & K_{\mathbf{x}} \in \mathcal{H}, \\ \forall (\mathbf{x}, f) \in \mathcal{X} \times \mathcal{H}, & \langle f, K_{\mathbf{x}} \rangle = f(\mathbf{x}). \end{cases}$$

- $\text{Ker}(K) = \text{Ker}(L^*) = \text{Ker}(L)$ , implying  $K\mathbf{1} = 0$ . Therefore, each row/column of  $K$  is in  $\mathcal{H}$ .
- For any  $f \in \mathcal{H}$ , if we note  $g_i = \langle K(i, \cdot), f \rangle$  we get:

$$g = KLf = L^*Lf = \Pi_{\mathcal{H}}(f) = f.$$

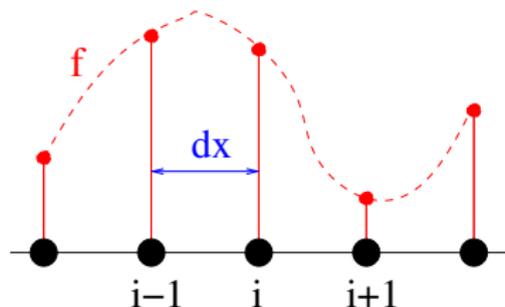
As a conclusion  $K = L^*$  is the reproducing kernel of  $\mathcal{H}$ .  $\square$

## Example



$$L^* = \begin{pmatrix} 0.88 & -0.12 & 0.08 & -0.32 & -0.52 \\ -0.12 & 0.88 & 0.08 & -0.32 & -0.52 \\ 0.08 & 0.08 & 0.28 & -0.12 & -0.32 \\ -0.32 & -0.32 & -0.12 & 0.48 & 0.28 \\ -0.52 & -0.52 & -0.32 & 0.28 & 1.08 \end{pmatrix}$$

# Interpretation of the Laplacian



$$\begin{aligned}\Delta f(x) &= f''(x) \\ &\sim \frac{f'(x + dx/2) - f'(x - dx/2)}{dx} \\ &\sim \frac{f(x + dx) - f(x) - f(x) + f(x - dx)}{dx^2} \\ &= \frac{f_{i-1} + f_{i+1} - 2f(x)}{dx^2} \\ &= -\frac{Lf(i)}{dx^2}.\end{aligned}$$

## Interpretation of regularization

For  $f = [0, 1] \rightarrow \mathbb{R}$  and  $x_i = i/m$ , we have:

$$\begin{aligned}\Omega(f) &= \sum_{i=1}^m \left( f\left(\frac{i+1}{m}\right) - f\left(\frac{i}{m}\right) \right)^2 \\ &\sim \sum_{i=1}^m \left( \frac{1}{m} \times f'\left(\frac{i}{m}\right) \right)^2 \\ &= \frac{1}{m} \times \frac{1}{m} \sum_{i=1}^m f'\left(\frac{i}{m}\right)^2 \\ &\sim \frac{1}{m} \int_0^1 f'(t)^2 dt.\end{aligned}$$

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - **Kernels on graphs**
    - Motivation
    - Graph distance and p.d. kernels
    - Construction by regularization
    - **The diffusion kernel**
    - Harmonic analysis on graphs
    - Applications

# Motivation

- Consider the normalized Gaussian kernel on  $\mathbb{R}^d$ :

$$K_t(\mathbf{x}, \mathbf{x}') = \frac{1}{(4\pi t)^{\frac{d}{2}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{4t}\right).$$

- In order to transpose it to the graph, replacing the Euclidean distance by the shortest-path distance does not work.
- In this section we provide a characterization of the Gaussian kernel as the **solution of a partial differential equation** involving the Laplacian, which we can transpose to the graph: the **diffusion equation**.
- The solution of the discrete diffusion equation will be called the **diffusion kernel** or **heat kernel**.

# The diffusion equation

## Lemma

For any  $\mathbf{x}_0 \in \mathbb{R}^d$ , the function:

$$K_{\mathbf{x}_0}(\mathbf{x}, t) = K_t(\mathbf{x}_0, \mathbf{x}) = \frac{1}{(4\pi t)^{\frac{d}{2}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{4t}\right).$$

is solution of the *diffusion equation*:

$$\frac{\partial}{\partial t} K_{\mathbf{x}_0}(\mathbf{x}, t) = \Delta K_{\mathbf{x}_0}(\mathbf{x}, t).$$

with initial condition  $K_{\mathbf{x}_0}(\mathbf{x}, 0) = \delta_{\mathbf{x}_0}(\mathbf{x})$

(proof = direct computation).

## Discrete diffusion equation

For finite-dimensional  $f_t \in \mathbb{R}^m$ , the diffusion equation becomes:

$$\frac{\partial}{\partial t} f_t = -L f_t$$

which admits the following solution:

$$f_t = f_0 e^{-tL}$$

with

$$e^{tL} = I - tL + \frac{t^2}{2!} L^2 - \frac{t^3}{3!} L^3 + \dots$$

## Diffusion kernel (Kondor and Lafferty, 2002)

This suggest to consider:

$$K = e^{-tL}$$

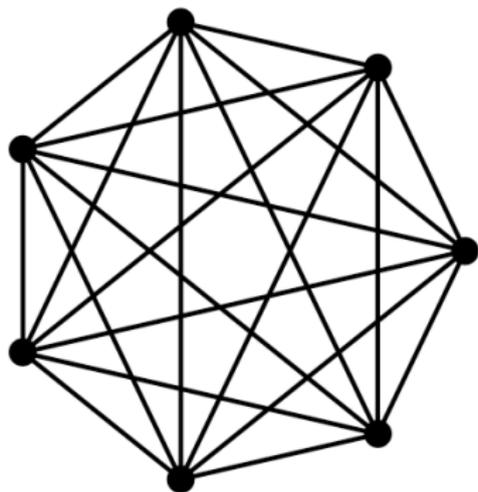
which is indeed symmetric positive semi-definite because if we write:

$$L = \sum_{i=1}^m \lambda_i u_i u_i^T \quad (\lambda_i \geq 0)$$

we obtain:

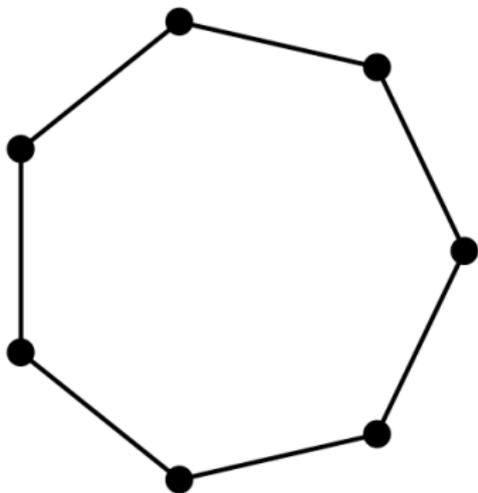
$$K = e^{-tL} = \sum_{i=1}^m e^{-t\lambda_i} u_i u_i^T$$

## Example: complete graph



$$K_{i,j} = \begin{cases} \frac{1+(m-1)e^{-tm}}{m} & \text{for } i = j, \\ \frac{1-e^{-tm}}{m} & \text{for } i \neq j. \end{cases}$$

## Example: closed chain



$$K_{i,j} = \frac{1}{m} \sum_{\nu=0}^{m-1} \exp \left[ -2t \left( 1 - \cos \frac{2\pi\nu}{m} \right) \right] \cos \frac{2\pi\nu(i-j)}{m}.$$

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - **Kernels on graphs**
    - Motivation
    - Graph distance and p.d. kernels
    - Construction by regularization
    - The diffusion kernel
    - **Harmonic analysis on graphs**
    - Applications

# Motivation

- In this section we show that the diffusion and Laplace kernels can be interpreted in the **frequency domain** of functions
- This shows that our strategy to design kernels on graphs was based on **(discrete) harmonic analysis** on the graph
- This follows the approach we developed for semigroup kernels!

## Spectrum of the diffusion kernel

- Let  $0 = \lambda_1 < \lambda_2 \leq \dots \leq \lambda_m$  be the eigenvalues of the Laplacian:

$$L = \sum_{i=1}^m \lambda_i u_i u_i^\top \quad (\lambda_i \geq 0)$$

- The diffusion kernel  $K_t$  is an **invertible** matrix because its eigenvalues are strictly positive:

$$K_t = \sum_{i=1}^m e^{-t\lambda_i} u_i u_i^\top$$

## Norm in the diffusion RKHS

- Any function  $f \in \mathbb{R}^m$  can be written as  $f = K (K^{-1}f)$ , therefore its norm in the diffusion RKHS is:

$$\|f\|_{K_t}^2 = (f^\top K^{-1}) K (K^{-1}f) = f^\top K^{-1}f.$$

- For  $i = 1, \dots, m$ , let:

$$\hat{f}_i = u_i^\top f$$

be the projection of  $f$  onto the eigenbasis of  $K$ .

- We then have:

$$\|f\|_{K_t}^2 = f^\top K^{-1}f = \sum_{i=1}^m e^{t\lambda_i} \hat{f}_i^2.$$

- This looks similar to  $\int |\hat{f}(\omega)|^2 e^{\sigma^2 \omega^2} d\omega \dots$

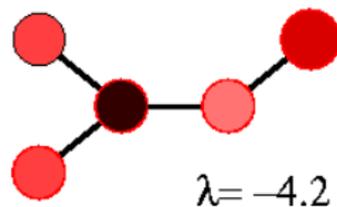
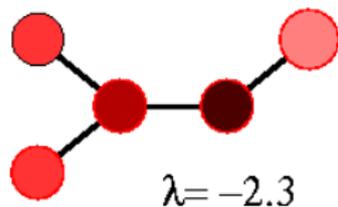
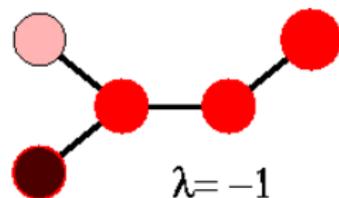
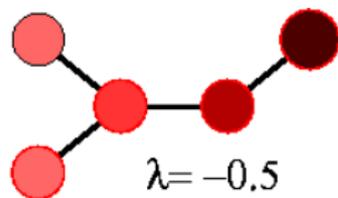
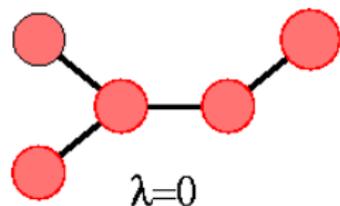
# Discrete Fourier transform

## Definition

The vector  $\hat{f} = (\hat{f}_1, \dots, \hat{f}_m)^\top$  is called the **discrete Fourier transform** of  $f \in \mathbb{R}^n$

- The eigenvectors of the Laplacian are the discrete equivalent to the sine/cosine Fourier basis on  $\mathbb{R}^n$ .
- The eigenvalues  $\lambda_j$  are the equivalent to the frequencies  $\omega^2$
- Successive eigenvectors “oscillate” increasingly as eigenvalues get more and more negative.

## Example: eigenvectors of the Laplacian



# Generalization

This observation suggests to define a whole family of kernels:

$$K_r = \sum_{i=1}^m r(\lambda_i) u_i u_i^\top$$

associated with the following RKHS norms:

$$\|f\|_{K_r}^2 = \sum_{i=1}^m \frac{\hat{f}_i^2}{r(\lambda_i)}$$

where  $r : \mathbb{R}^+ \rightarrow \mathbb{R}_*^+$  is a **non-increasing** function.

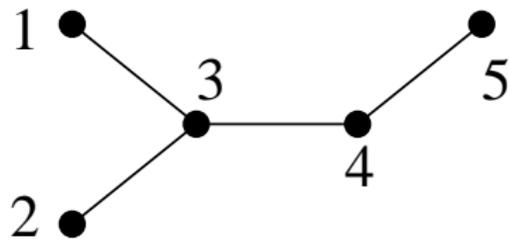
## Example : regularized Laplacian

$$r(\lambda) = \frac{1}{\lambda + \epsilon}, \quad \epsilon > 0$$

$$K = \sum_{i=1}^m \frac{1}{\lambda_i + \epsilon} u_i u_i^\top = (L + \epsilon I)^{-1}$$

$$\|f\|_K^2 = f^\top K^{-1} f = \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 + \epsilon \sum_{i=1}^m f(\mathbf{x}_i)^2.$$

## Example



$$(L + I)^{-1} = \begin{pmatrix} 0.60 & 0.10 & 0.19 & 0.08 & 0.04 \\ 0.10 & 0.60 & 0.19 & 0.08 & 0.04 \\ 0.19 & 0.19 & 0.38 & 0.15 & 0.08 \\ 0.08 & 0.08 & 0.15 & 0.46 & 0.23 \\ 0.04 & 0.04 & 0.08 & 0.23 & 0.62 \end{pmatrix}$$

# Outline

- 4 The Kernel Jungle
  - Mercer kernels and shift-invariant kernels
  - Kernels for generative models
  - Kernels for biological sequences
  - Kernels for graphs
  - **Kernels on graphs**
    - Motivation
    - Graph distance and p.d. kernels
    - Construction by regularization
    - The diffusion kernel
    - Harmonic analysis on graphs
    - Applications

# Applications 1: graph partitioning

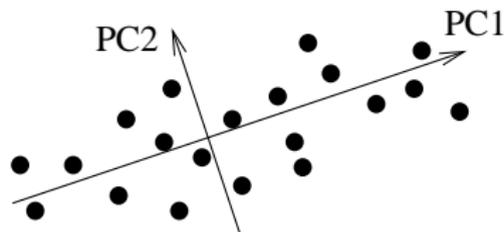
- A classical relaxation of graph partitioning is:

$$\min_{f \in \mathbb{R}^{\mathcal{X}}} \sum_{i \sim j} (f_i - f_j)^2 \quad \text{s.t.} \quad \sum_i f_i^2 = 1$$

- This can be rewritten

$$\max_f \sum_i f_i^2 \quad \text{s.t.} \quad \|f\|_{\mathcal{H}} \leq 1$$

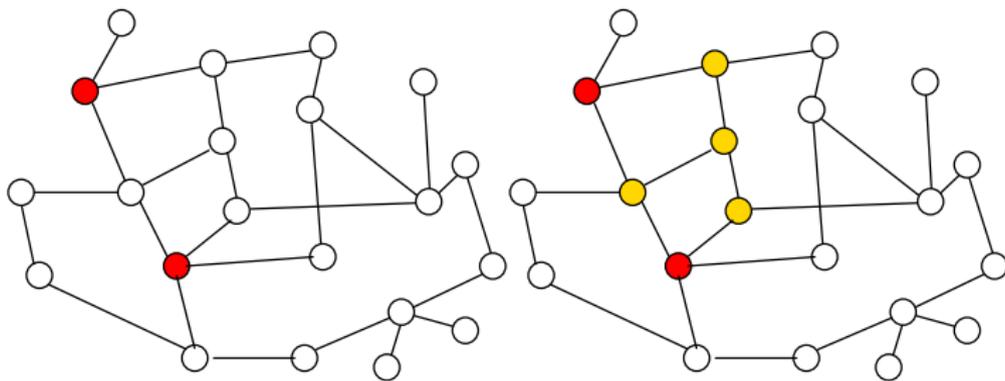
- This is **principal component analysis** in the RKHS (“kernel PCA”)



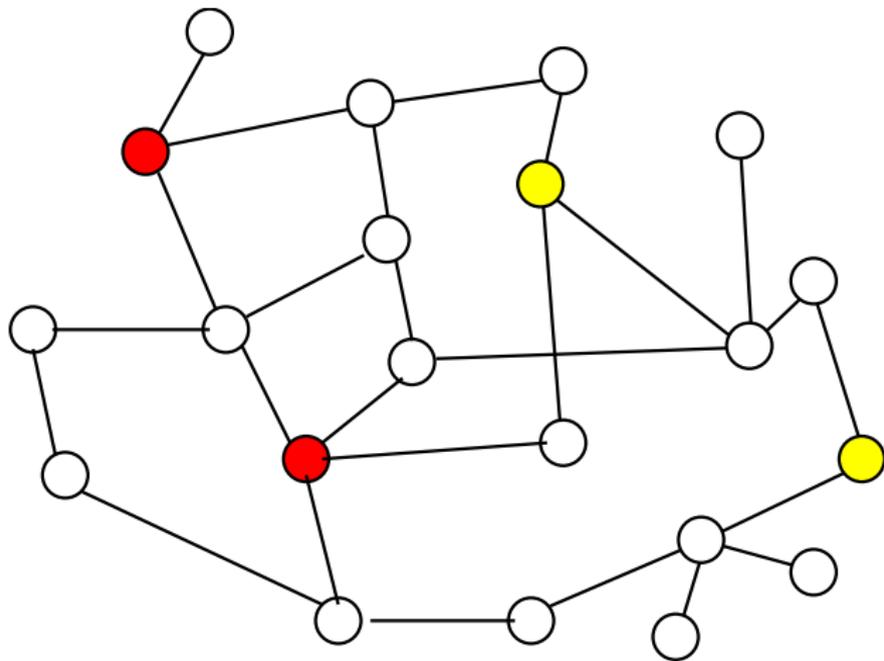
## Applications 2: search on a graph

- Let  $x_1, \dots, x_q$  a set of  $q$  nodes (the **query**). How to find “similar” nodes (and rank them)?
- One solution:

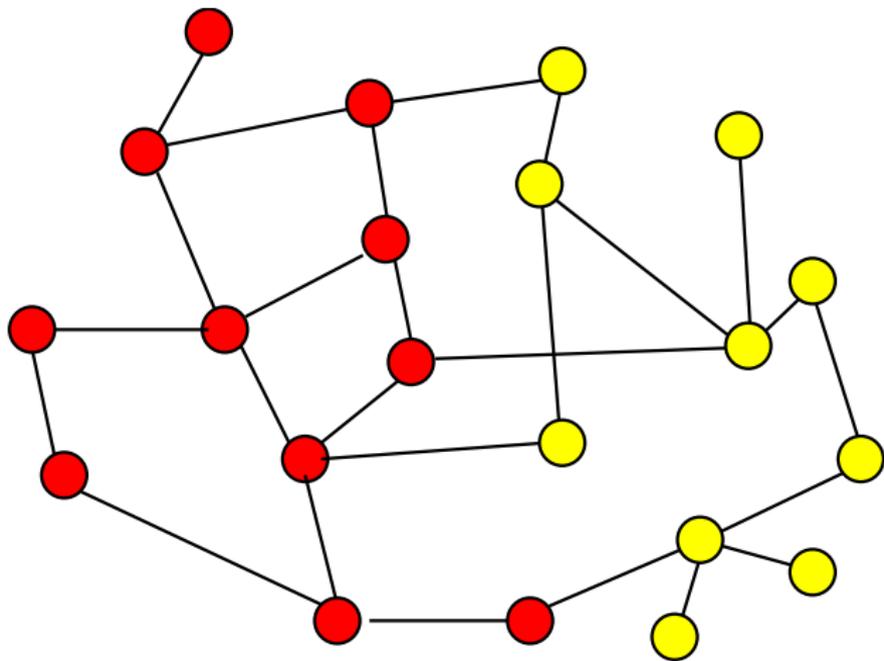
$$\min_f \|f\|_{\mathcal{H}} \quad \text{s.t.} \quad f(x_i) \geq 1 \text{ for } i = 1, \dots, q.$$



## Application 3: Semi-supervised learning



## Application 3: Semi-supervised learning



## Application 4: Tumor classification from microarray data (Rapaport et al., 2006)

### Data available

- Gene expression measures for **more than 10k genes**
- Measured on **less than 100 samples** of two (or more) different classes (e.g., different tumors)

# Application 4: Tumor classification from microarray data (Rapaport et al., 2006)

## Data available

- Gene expression measures for **more than 10k genes**
- Measured on **less than 100 samples** of two (or more) different classes (e.g., different tumors)

## Goal

- Design a **classifier** to automatically assign a class to future samples from their expression profile
- **Interpret** biologically the differences between the classes

# Linear classifiers

## The approach

- Each sample is represented by a vector  $x = (x_1, \dots, x_p)$  where  $p > 10^5$  is the number of probes
- **Classification**: given the set of labeled sample, learn a linear decision function:

$$f(x) = \sum_{i=1}^p \beta_i x_i + \beta_0 ,$$

that is positive for one class, negative for the other

- **Interpretation**: the weight  $\beta_i$  quantifies the influence of gene  $i$  for the classification

# Linear classifiers

## Pitfalls

- **No robust estimation procedure** exist for 100 samples in  $10^5$  dimensions!
- It is necessary to **reduce the complexity** of the problem with **prior knowledge**.

# Example : Norm Constraints

## The approach

A common method in statistics to learn with few samples in high dimension is to **constrain the norm of  $\beta$** , e.g.:

- Euclidean norm (support vector machines, ridge regression):  
$$\|\beta\|_2 = \sum_{i=1}^p \beta_i^2$$
- $L_1$ -norm (lasso regression) :  $\|\beta\|_1 = \sum_{i=1}^p |\beta_i|$

## Pros

- Good performance in classification

## Cons

- Limited interpretation (small weights)
- No prior biological knowledge

## Example 2: Feature Selection

### The approach

Constrain most weights to be 0, i.e., **select a few genes** ( $< 20$ ) whose expression are enough for classification. Interpretation is then about the selected genes.

### Pros

- Good performance in classification
- Useful for **biomarker** selection
- Apparently easy interpretation

### Cons

- The gene selection process is usually **not robust**
- Wrong interpretation is the rule (too much correlation between genes)

# Pathway interpretation

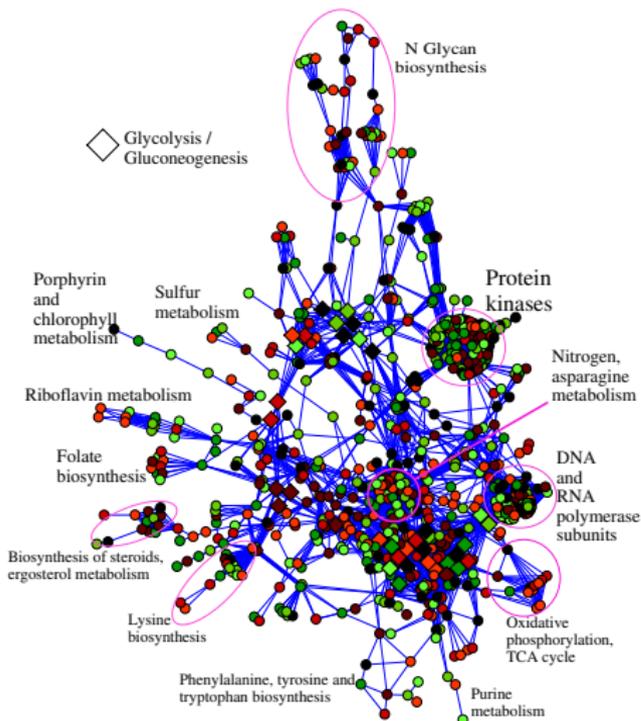
## Motivation

- Basic biological functions are usually expressed in terms of **pathways** and not of single genes (metabolic, signaling, regulatory)
- Many pathways are already known
- How to use this prior knowledge to **constrain the weights to have an interpretation at the level of pathways?**

## Solution (Rapaport et al., 2006)

- **Constrain the diffusion RKHS norm of  $\beta$**
- Relevant if the true decision function is indeed smooth w.r.t. the biological network

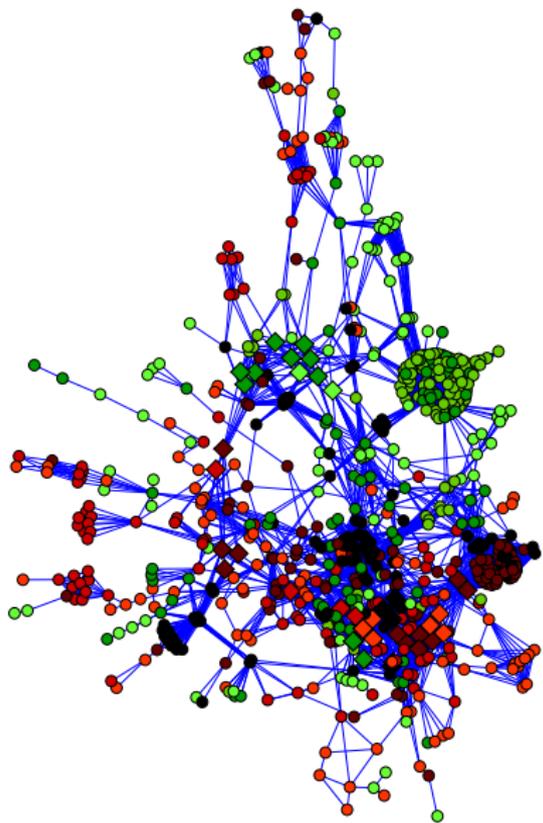
# Pathway interpretation



## Bad example

- The graph is the complete known **metabolic network** of the budding yeast (from KEGG database)
- We project the **classifier weight** learned by a SVM
- Good classification accuracy, but **no possible interpretation!**

# Pathway interpretation



## Good example

- The graph is the complete known **metabolic network** of the budding yeast (from KEGG database)
- We project the **classifier weight** learned by a spectral SVM
- Good classification accuracy, **and good interpretation!**

# References I

- N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68:337 – 404, 1950. URL <http://www.jstor.org/stable/1990404>.
- C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic analysis on semigroups*. Springer-Verlag, New-York, 1984.
- K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 74–81, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2278-5. doi: <http://dx.doi.org/10.1109/ICDM.2005.132>.
- S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.
- M. Cuturi and J.-P. Vert. The context-tree kernel for strings. *Neural Network.*, 18(4): 1111–1123, 2005. doi: 10.1016/j.neunet.2005.07.010. URL <http://dx.doi.org/10.1016/j.neunet.2005.07.010>.
- M. Cuturi, K. Fukumizu, and J.-P. Vert. Semigroup kernels on measures. *J. Mach. Learn. Res.*, 6:1169–1198, 2005. URL <http://jmlr.csail.mit.edu/papers/v6/cuturi05a.html>.

## References II

- T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: hardness results and efficient alternatives. In B. Schölkopf and M. Warmuth, editors, *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory and the Seventh Annual Workshop on Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143, Heidelberg, 2003. Springer. doi: 10.1007/b12006. URL <http://dx.doi.org/10.1007/b12006>.
- Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, pages 1–8. IEEE Computer Society, 2007. doi: 10.1109/CVPR.2007.383049. URL <http://dx.doi.org/10.1109/CVPR.2007.383049>.
- D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz, 1999.
- C. Helma, T. Cramer, S. Kramer, and L. De Raedt. Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. *J. Chem. Inf. Comput. Sci.*, 44(4): 1402–11, 2004. doi: 10.1021/ci034254q. URL <http://dx.doi.org/10.1021/ci034254q>.
- T. Jaakkola, M. Diekhans, and D. Haussler. A Discriminative Framework for Detecting Remote Protein Homologies. *J. Comput. Biol.*, 7(1,2):95–114, 2000. URL <http://www.cse.ucsc.edu/research/compbio/discriminative/Jaakola2-1998.ps>.

## References III

- T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proc. of Tenth Conference on Advances in Neural Information Processing Systems*, 1999. URL <http://www.cse.ucsc.edu/research/ml/papers/Jaakkola.ps>.
- H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In T. Faucett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328, New York, NY, USA, 2003. AAAI Press.
- T. Kin, K. Tsuda, and K. Asai. Marginalized kernels for RNA sequence data analysis. In R. Lathtop, K. Nakai, S. Miyano, T. Takagi, and M. Kanehisa, editors, *Genome Informatics 2002*, pages 112–122. Universal Academic Press, 2002. URL <http://www.jsbi.org/journal/GIW02/GIW02F012.html>.
- R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *J. Mach. Learn. Res.*, 5:1435–1455, 2004.
- C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: a string kernel for SVM protein classification. In R. B. Altman, A. K. Dunker, L. Hunter, K. Lauerdale, and T. E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing 2002*, pages 564–575, Singapore, 2002. World Scientific.

## References IV

- L. Liao and W. Noble. Combining Pairwise Sequence Similarity and Support Vector Machines for Detecting Remote Protein Evolutionary and Structural Relationships. *J. Comput. Biol.*, 10(6):857–868, 2003. URL <http://www.liebertonline.com/doi/abs/10.1089/106652703322756113>.
- H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. n. p. v. d. d. r. Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, 2002. URL <http://www.ai.mit.edu/projects/jmlr/papers/volume2/lodhi02a/abstract.html>.
- B. Logan, P. Moreno, B. Suzek, Z. Weng, and S. Kasif. A Study of Remote Homology Detection. Technical Report CRL 2001/05, Compaq Cambridge Research laboratory, June 2001.
- P. Mahé and J. P. Vert. Graph kernels based on tree patterns for molecules. *Mach. Learn.*, 75(1):3–35, 2009. doi: 10.1007/s10994-008-5086-2. URL <http://dx.doi.org/10.1007/s10994-008-5086-2>.
- P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*, pages 552–559. ACM Press, 2004.
- P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *J. Chem. Inf. Model.*, 45(4):939–51, 2005. doi: 10.1021/ci050039t. URL <http://dx.doi.org/10.1021/ci050039t>.

## References V

- A. Nicholls. Oechem, version 1.3.4, openeye scientific software. website, 2005.
- F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In T. Washio and L. De Raedt, editors, *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74, 2003.
- F. Rapaport, A. Zynoviev, M. Dutreix, E. Barillot, and J.-P. Vert. Classification of microarray data using gene networks. *BMC Bioinformatics*, 8:35, 2007. doi: 10.1186/1471-2105-8-35. URL <http://dx.doi.org/10.1186/1471-2105-8-35>.
- H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004. URL <http://bioinformatics.oupjournals.org/cgi/content/abstract/20/11/1682>.
- M. Seeger. Covariance Kernels from Bayesian Generative Models. In *Adv. Neural Inform. Process. Syst.*, volume 14, pages 905–912, 2002.
- J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.
- N. Sherashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 488–495, Clearwater Beach, Florida USA, 2009. Society for Artificial Intelligence and Statistics.

## References VI

- T. Smith and M. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.
- K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414, 2002a. doi: 10.1162/08997660260293274. URL <http://dx.doi.org/10.1162/08997660260293274>.
- K. Tsuda, T. Kin, and K. Asai. Marginalized Kernels for Biological Sequences. *Bioinformatics*, 18:S268–S275, 2002b.
- J.-P. Vert, H. Saigo, and T. Akutsu. Local alignment kernels for biological sequences. In B. Schölkopf, K. Tsuda, and J. Vert, editors, *Kernel Methods in Computational Biology*, pages 131–154. MIT Press, The MIT Press, Cambridge, Massachusetts, 2004.
- J.-P. Vert, R. Thurman, and W. S. Noble. Kernels for gene regulatory regions. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Adv. Neural. Inform. Process Syst.*, volume 18, pages 1401–1408, Cambridge, MA, 2006. MIT Press.