Machine Learning with Kernel Methods

Julien Mairal & Jean-Philippe Vert

firstname.lastname@m4x.org

Last update: Dec 2020
Starting point: what we know is how to solve
Or
But real data are often more complicated...
Main goal of this course

Extend well-understood, linear statistical learning techniques to real-world, complicated, structured, high-dimensional data based on a rigorous mathematical framework leading to practical modelling tools and algorithms.
Organization of the course

Contents

1. Present the basic mathematical theory of kernel methods.
2. Introduce algorithms for supervised and unsupervised machine learning with kernels.
3. Develop a working knowledge of kernel engineering for specific data and applications (graphs, biological sequences, images).
4. Discuss open research topics related to kernels such as large-scale learning with kernels and “deep kernel learning”.

Practical

- Course homepage with slides, schedules, homework etc...:
  http://cbio.mines-paristech.fr/~jvert/svn/kernelcourse/course/2020mva
- Evaluation: 40% homework + 60% final exam.
Outline

1. Kernels and RKHS
   - Positive Definite Kernels
   - Reproducing Kernel Hilbert Spaces (RKHS)
   - Examples
   - Smoothness functional
Outline

1 Kernels and RKHS
   • Positive Definite Kernels
   • Reproducing Kernel Hilbert Spaces (RKHS)
   • Examples
   • Smoothness functional

2 Kernel tricks
   • The kernel trick
   • The representer theorem
Outline

1. Kernels and RKHS
   - Positive Definite Kernels
   - Reproducing Kernel Hilbert Spaces (RKHS)
   - Examples
   - Smoothness functional

2. Kernel tricks
   - The kernel trick
   - The representer theorem

3. Kernel Methods: Supervised Learning
   - Kernel ridge regression
   - Kernel logistic regression
   - Large-margin classifiers
   - Interlude: convex optimization and duality
   - Support vector machines
Outline

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

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

5 The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic models
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs
Outline

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

5 The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic models
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs

6 Open Problems and Research Topics
- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
- Foundations of deep learning from a kernel point of view
Kernels and RKHS
Overview

Motivations
- Develop **versatile** algorithms to process and analyze data...
- ...without making any assumptions 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
   - Positive Definite Kernels
   - Reproducing Kernel Hilbert Spaces (RKHS)
   - Examples
   - Smoothness functional

2. Kernel tricks

3. Kernel Methods: Supervised Learning


5. The Kernel Jungle

6. Open Problems and Research Topics
Representation by pairwise comparisons

\[
\phi(S) = (\text{aatcgagtcac, atggacgtct, tgcactact})
\]

\[
K = \begin{pmatrix}
1 & 0.5 & 0.3 \\
0.5 & 1 & 0.6 \\
0.3 & 0.6 & 1 \\
\end{pmatrix}
\]

**Idea**

- Define a “comparison function”: \( K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R} \).
- Represent a set of \( n \) data points \( S = \{x_1, x_2, \ldots, x_n\} \) by the \( n \times n \) matrix:

\[
[K]_{ij} := K(x_i, x_j).
\]
Representation by pairwise comparisons

Remarks

- **K** is always an $n \times n$ matrix, whatever the nature of data: the same algorithm will work for any type of data (vectors, strings, ...).
- Total modularity between the choice of function $K$ and the choice of the algorithm.
- Poor scalability with respect to the dataset size ($n^2$ to compute and store $K$)... but wait until the end of the course to see how to deal with large-scale problems.
- 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 a set $\mathcal{X}$ is a function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ that is symmetric:

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

and which satisfies, for all $N \in \mathbb{N}$, $(x_1, x_2, \ldots, x_N) \in \mathcal{X}^N$ and $(a_1, a_2, \ldots, a_N) \in \mathbb{R}^N$:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} a_ia_j K(x_i, x_j) \geq 0.$$
Equivalently, a kernel $K$ is p.d. if and only if, for any $N \in \mathbb{N}$ and any set of points $(x_1, x_2, \ldots, x_N) \in \mathcal{X}^N$, the similarity matrix $[K]_{ij} := K(x_i, x_j)$ is positive semidefinite.

Kernel methods are algorithms that take such matrices as input.
The simplest p.d. kernel, for real numbers

**Lemma**

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

$$\forall (x, x') \in \mathbb{R}^2, \quad K(x, x') = xx'$$

is p.d.
The simplest p.d. kernel, for real numbers

Lemma

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

$$\forall (x, x') \in \mathbb{R}^2, \quad K(x, x') = xx'$$

is p.d.

Proof:

1. $xx' = x'x$
2. $\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j x_i x_j = \left( \sum_{i=1}^{N} a_i x_i \right)^2 \geq 0$
The simplest p.d. kernel, for vectors

**Lemma**

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

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

is p.d. (it is often called the linear kernel).
The simplest p.d. kernel, for vectors

**Lemma**

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

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

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

**Proof:**

\[
\begin{align*}
\langle x, x' \rangle_{\mathbb{R}^d} &= \langle x', x \rangle_{\mathbb{R}^d} \\
\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j \langle x_i, x_j \rangle_{\mathbb{R}^d} &= \left\| \sum_{i=1}^{N} a_i x_i \right\|_{\mathbb{R}^d}^2 \geq 0
\end{align*}
\]

\(\square\)
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.:

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

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

Proof:

- $\langle \Phi(x), \Phi(x') \rangle_{\mathbb{R}^d} = \langle \Phi(x'), \Phi(x) \rangle_{\mathbb{R}^d}$
- $\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathbb{R}^d} = \| \sum_{i=1}^{N} a_i \Phi(x_i) \|_{\mathbb{R}^d}^2 \geq 0$
Example: polynomial kernel

For $\mathbf{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\Phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$: 
Example: polynomial kernel

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

\[
K(x, x') = x_1^2x_1'^2 + 2x_1x_2x_1'^x_2' + x_2^2x_2'^2
= (x_1x_1' + x_2x_2')^2
= \left< x, x' \right>_{\mathbb{R}^2}^2.
\]
Example: polynomial kernel

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

\[
K(\mathbf{x}, \mathbf{x}') = x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2 x_2'^2 \\
= (x_1x_1' + x_2x_2')^2 \\
= \langle \mathbf{x}, \mathbf{x}' \rangle^2_{\mathbb{R}^2}.
\]

Exercise: show that \( \langle \mathbf{x}, \mathbf{x}' \rangle^d_{\mathbb{R}^p} \) is p.d. on \( \mathcal{X} = \mathbb{R}^p \) for any \( d \in \mathbb{N} \).
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 $x, x'$ in $\mathcal{X}$:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}.$$
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 $\langle f, f \rangle_{\mathcal{H}} > 0$ 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 Cauchy sequence $(f_n)_{n \geq 0}$ is a sequence whose elements become progressively arbitrarily close to each other:

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

A Hilbert space is a pre-Hilbert space complete for the norm $\| . \|_{\mathcal{H}}$. That is, any Cauchy sequence in $\mathcal{H}$ converges in $\mathcal{H}$.

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

- Assume $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is finite of size $N$.
- Any p.d. kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is entirely defined by the $N \times N$ symmetric positive semidefinite matrix $[K]_{ij} := K(x_i, x_j)$.
- It can therefore be diagonalized on an orthonormal basis of eigenvectors $(u_1, u_2, \ldots, u_N)$, with non-negative eigenvalues $0 \leq \lambda_1 \leq \ldots \leq \lambda_N$, i.e.,

$$K(x_i, x_j) = \left[ \sum_{l=1}^{N} \lambda_l u_l u_l^\top \right]_{ij} = \sum_{l=1}^{N} \lambda_l [u_l]_i [u_l]_j = \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathbb{R}^N},$$

with

$$\Phi(x_i) = \begin{pmatrix} \sqrt{\lambda_1}[u_1]_i \\ \vdots \\ \sqrt{\lambda_N}[u_N]_i \end{pmatrix}.$$
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
   - Positive Definite Kernels
   - Reproducing Kernel Hilbert Spaces (RKHS)
   - Examples
   - Smoothness functional

2. Kernel tricks

3. Kernel Methods: Supervised Learning


5. The Kernel Jungle

6. Open Problems and Research Topics
Before we go into formal details

- Among the Hilbert spaces $\mathcal{H}$ mentioned in Aronszjan’s theorem, we will see that one of them, called RKHS, is of interest to us.
- This is a space of functions from $\mathcal{X}$ to $\mathbb{R}$.
- In other words, each data point $x$ in $\mathcal{X}$ will be represented by a function $\Phi(x) = K_x$ in $\mathcal{H}$.

**Example of functional mapping**

Consider $\mathcal{X} = \mathbb{R}$. We could decide to represent each scalar $x$ in $\mathbb{R}$ as a Gaussian function centered at $x$:

$$K_x(y) \mapsto e^{-\frac{1}{2} \alpha (x-y)^2}.$$
Functional spaces for machine learning

Before we go into formal details

- Among the Hilbert spaces $\mathcal{H}$ mentioned in Aronszjan’s theorem, we will see that one of them, called RKHS, is of interest to us.
- This is a space of functions from $\mathcal{X}$ to $\mathbb{R}$.
- In other words, each data point $x$ in $\mathcal{X}$ will be represented by a function $\Phi(x) = K_x$ in $\mathcal{H}$.

Example of functional mapping

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

  $$K_x : y \mapsto e^{-\frac{1}{2\alpha}(x-y)^2}.$$  

- What would be the corresponding $\mathcal{H}$ (if it exists)? What would be the inner-product?
Functional spaces for machine learning

What does it mean to map a data point to a function?

Ex: if \( x, y \) in \( \mathbb{R} \) and \( K(x, y) = e^{-\frac{1}{\sigma^2}(x-y)^2} \) is the Gaussian kernel,

\[
\Phi(x) : t \mapsto e^{-\frac{1}{2\alpha^2}(x-t)^2}
\]

\[
\Phi(y) : t \mapsto e^{-\frac{1}{2\alpha^2}(y-t)^2}
\]

- Data points are mapped to Gaussian functions living in a Hilbert space \( \mathcal{H} \).
- But \( \mathcal{H} \) is much richer and contains much more than Gaussian functions!
- Prediction functions \( f \) live in \( \mathcal{H} \): \( f(x) = \langle f, \Phi(x) \rangle \).
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

\[ \forall x \in \mathcal{X}, \quad K_x : t \mapsto K(x, t). \]

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

\[ f(x) = \langle f, K_x \rangle_{\mathcal{H}}. \]

If a r.k. exists, then $\mathcal{H}$ is called a reproducing kernel Hilbert space (RKHS).
RKHS: why do we care?

The principle of RKHS gives us a simple recipe to do machine learning:

- Map data \( x \) in \( \mathcal{X} \) to a high-dimensional Hilbert space \( \mathcal{H} \) (the RKHS) through a kernel mapping \( \Phi : \mathcal{X} \rightarrow \mathcal{H} \), with \( \Phi(x) = K_x \).
- In \( \mathcal{H} \), consider simple linear models \( f(x) = \langle f, \Phi(x) \rangle_{\mathcal{H}} \).
- If \( \mathcal{X} = \mathbb{R}^p \), a linear function in \( \Phi(x) \) may be nonlinear in \( x \).
- For instance, for supervised learning, given training data \( (y_i, x_i)_{i=1,...,n} \), we may want to minimize the empirical risk.

\[
\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \| f \|^2_{\mathcal{H}}.
\]
RKHS: why do we care?

The principle of RKHS gives us a simple recipe to do machine learning:

- Map data $x$ in $\mathcal{X}$ to a high-dimensional Hilbert space $\mathcal{H}$ (the RKHS) through a kernel mapping $\Phi : \mathcal{X} \rightarrow \mathcal{H}$, with $\Phi(x) = K_x$.
- In $\mathcal{H}$, consider simple linear models $f(x) = \langle f, \Phi(x) \rangle_\mathcal{H}$.
- If $\mathcal{X} = \mathbb{R}^p$, a linear function in $\Phi(x)$ may be nonlinear in $x$.
- For instance, for supervised learning, given training data $(y_i, x_i)_{i=1,...,n}$, we may want to minimize the empirical risk.

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

More formal details to come...
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 $x \in \mathcal{X}$, the (linear) mapping:

$$F : \mathcal{H} \to \mathbb{R}$$

$$f \mapsto f(x)$$

is continuous.
An equivalent definition of RKHS

**Theorem**

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

$$F : \mathcal{H} \rightarrow \mathbb{R}$$

$$f \mapsto f(x)$$

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 $(f_n(x))_{n \in \mathbb{N}}$ converges to $f(x)$ for any $x \in X$. 
Proof

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

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

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

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

Since $F$ is linear, it is indeed sufficient to show that $f \to 0 \Rightarrow f(x) \to 0$. 

Proof (Converse)

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

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

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

\[
 f(x) = \langle f, g_x \rangle_{\mathcal{H}}. 
\]

The function \( K(x, y) = g_x(y) \) is then a r.k. for \( \mathcal{H} \).
Uniqueness 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.
Uniqueness 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 $x \in \mathcal{X}$:

$$
\| K_x - K'_x \|_{\mathcal{H}}^2 = \langle K_x - K'_x, K_x - K'_x \rangle_{\mathcal{H}}
$$

$$
= \langle K_x - K'_x, K_x \rangle_{\mathcal{H}} - \langle K_x - K'_x, K'_x \rangle_{\mathcal{H}}
$$

$$
= K_x(x) - K'_x(x) - K_x(x) + K'_x(x)
$$

$$
= 0.
$$

This shows that $K_x = K'_x$ as functions, i.e., $K_x(y) = K'_x(y)$ for any $y \in \mathcal{X}$. In other words, $K = K'$. □
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 $x \in \mathcal{X}$:

$$
\| K_x - K'_x \|_H^2 = \langle K_x - K'_x, K_x - K'_x \rangle_H \\
= \langle K_x - K'_x, K_x \rangle_H - \langle K_x - K'_x, K'_x \rangle_H \\
= K_x(x) - K'_x(x) - K_x(x) + K'_x(x) \\
= 0.
$$

This shows that $K_x = K'_x$ as functions, i.e., $K_x(y) = K'_x(y)$ for any $y \in \mathcal{X}$. In other words, $K=K'$.

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 \((x, y) \in X^2:\)

\[ K(x, y) = \langle K_x, K_y \rangle_H = \langle K_y, K_x \rangle_H = K(y, x). \]

2 It is p.d. because for any \(N \in \mathbb{N}, (x_1, x_2, \ldots, x_N) \in X^N,\) and \((a_1, a_2, \ldots, a_N) \in \mathbb{R}^N:\)

\[
\sum_{i,j=1}^{N} a_i a_j K(x_i, x_j) = \sum_{i,j=1}^{N} a_i a_j \langle K_{x_i}, K_{x_j} \rangle_H \\
= \| \sum_{i=1}^{N} a_i K_{x_i} \|_H^2 \\
\geq 0. \quad \square
\]
Proof

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

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

$$f = \sum_{i=1}^{m} a_i K_{x_i}, \quad g = \sum_{j=1}^{n} b_j K_{y_j},$$

let:

$$\langle f, g \rangle_{\mathcal{H}_0} := \sum_{i,j} a_i b_j K(x_i, y_j).$$
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:

\[
\langle f, g \rangle_{\mathcal{H}_0} = \sum_{i=1}^{m} a_i g(x_i) = \sum_{j=1}^{n} b_j f(y_j).
\]

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

\[
\langle f, K_x \rangle_{\mathcal{H}_0} = f(x).
\]
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:

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

Therefore for any \(x\) the sequence \((f_n(x))_{n \geq 0}\) 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).  \(\square\)
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 \( x, x' \) in \( \mathcal{X} \):

\[ K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}. \]
Proof of Aronzsajn’s theorem

- 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 $\Phi : \mathcal{X} \to \mathcal{H}$ defined by:

\[ \forall x \in \mathcal{X}, \quad \Phi(x) = K_x. \]

- By the reproducing property we have:

\[ \forall (x, y) \in \mathcal{X}^2, \quad \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}} = \langle K_x, K_y \rangle_{\mathcal{H}} = K(x, y). \quad \square \]
Outline

1 Kernels and RKHS
   - Positive Definite Kernels
   - Reproducing Kernel Hilbert Spaces (RKHS)
   - Examples
     - Smoothness functional

2 Kernel tricks

3 Kernel Methods: Supervised Learning

4 Kernel Methods: Unsupervised Learning

5 The Kernel Jungle

6 Open Problems and Research Topics
The linear kernel

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

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

**Theorem**

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

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

endowed with the inner product

$$\forall w, v \in \mathbb{R}^d, \quad \langle f_w, f_v \rangle_{\mathcal{H}} = \langle w, v \rangle_{\mathbb{R}^d}$$

and corresponding norm

$$\forall w \in \mathbb{R}^d, \quad \| f_w \|_{\mathcal{H}} = \| w \|_2.$$
Proof

The set $\mathcal{H}$ of functions described in the theorem is the dual of $\mathbb{R}^d$, hence it is a Hilbert space:

$$\mathcal{H} = \left\{ f_w(x) = \langle w, x \rangle_{\mathbb{R}^d} : w \in \mathbb{R}^d \right\}.$$ 

- $\mathcal{H}$ contains all functions of the form $K_w : x \mapsto \langle w, x \rangle_{\mathbb{R}^d}$.
- For every $x$ in $\mathbb{R}^d$, and $f_w$ in $\mathcal{H}$,

$$f_w(x) = \langle w, x \rangle_{\mathbb{R}^d} = \langle f_w, K_x \rangle_{\mathcal{H}}.$$ 

$\mathcal{H}$ is thus the RKHS of the linear kernel.
Let us find the RKHS of the polynomial kernel of degree 2:

\[ \forall x, y \in \mathbb{R}^d, \quad K(x, y) = \langle x, y \rangle^2_{\mathbb{R}^d} = (x^T y)^2 \]
The polynomial kernel

Let us find the RKHS of the polynomial kernel of degree 2:

\[ \forall x, y \in \mathbb{R}^d, \quad K(x, y) = \langle x, y \rangle_{\mathbb{R}^d}^2 = \left( x^\top y \right)^2 \]

First step: Look for an inner-product.

\[
K(x, y) = \text{trace} \left( x^\top y x^\top y \right) \\
= \text{trace} \left( y^\top x x^\top y \right) \\
= \text{trace} \left( xx^\top yy^\top \right) \\
= \left\langle xx^\top, yy^\top \right\rangle_F,
\]

where \( F \) is the Froebenius norm for matrices in \( \mathbb{R}^{d \times d} \). Note that we have proven here that \( K \) is p.d.
The polynomial kernel

Second step: propose a candidate RKHS.
We know that $H$ contains all the functions

$$f(x) = \sum_i a_i K(x_i, x) = \sum_i a_i \left\langle x; x_i^\top, xx^\top \right\rangle_F$$

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

$$f_S(x) = \left\langle S, xx^\top \right\rangle_F = x^\top S x \quad \text{for} \quad S \in S^{d \times d},$$

where $S^{d \times d}$ is the set of symmetric matrices in $\mathbb{R}^{d \times d}$, endowed with the inner-product $\left\langle f_{S_1}, f_{S_2} \right\rangle_H = \left\langle S_1, S_2 \right\rangle_F$.

\[1\] Why is it important?
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, isomorphic to \( S^{d \times d} \) by \( \Phi : S \mapsto f_S \). Sometimes, things are not so simple and we need to prove the completeness explicitly.

**Fourth step: check that \( \mathcal{H} \) is the RKHS.**

1. \( \mathcal{H} \) contains all the functions \( K_x : t \mapsto K(x, t) = \langle xx^\top, tt^\top \rangle_F \).
2. For all \( f_S \) in \( \mathcal{H} \) and \( x \) in \( \mathcal{X} \),

\[
 f_S(x) = \left\langle S, xx^\top \right\rangle_F = \langle f_S, f_{xx^\top} \rangle_H = \langle f_S, K_x \rangle_H .
\]

\[\square\]

**Remark**

All points \( x \) in \( \mathcal{X} \) are mapped to a rank-one matrix \( xx^\top \), hence to a function \( K_x = f_{xx^\top} \) in \( \mathcal{H} \). However, most of points in \( \mathcal{H} \) do not admit a pre-image (why?).

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

Theorem

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

  $K_1 + K_2,$
  
  $K_1 K_2,$ and
  
  $cK_1,$ 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 $K$:

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

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

Proof: for $K_1 K_2,$ see next slide; otherwise, left as exercise
Proof for $K_1 K_2$ is p.d.

Proof.

Consider $n$ points in $\mathcal{X}$ and the corresponding $n \times n$ p.s.d. kernel matrices $K_1$ and $K_2$. As p.s.d. matrices, they admit factorizations $K_1 = X^T X$ and $K_2 = Y^T Y$. Then,

$$[K]_{ij} = [K_1]_{ij} [K_2]_{ij}$$

$$= \text{trace} \left( (x_i^T x_j)(y_j^T y_i) \right)$$

$$= \text{trace} \left( (y_j x_i^T)(x_j y_i^T) \right)$$

$$= \langle x_i y_i^T, x_j y_j^T \rangle_F.$$ 

$$= \langle z_i, z_j \rangle_{\mathbb{R}^{n^2}},$$

where the $x_i$’s and the $y_i$’s are the columns of $X$ and $Y$, respectively and $z_i = \text{vec}(x_i y_i^T)$. Thus, $K$ is p.s.d. and $K = K_1 K_2$ is a p.d. kernel. \qed
Examples

Theorem

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

**Theorem**

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

Proof:

$$e^K(x,x') = \lim_{n \to +\infty} \sum_{i=0}^{n} \frac{K(x, x')^i}{i!}$$
Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
Quizz: which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
Quizz: which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \log(1 + xx')$
Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \log (1 + xx')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \exp (-|x - x'|^2)$
Quizz: which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \log (1 + xx')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \exp (-|x - x'|^2)$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos (x + x')$
Quizz: which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \log(1 + xx')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \exp(-|x - x'|^2)$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos(x + x')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos(x - x')$
Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \log (1 + xx')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \exp (-|x - x'|^2)$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos (x + x')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos (x - x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \min(x, x')$
Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K (x, x') = \frac{1}{1 - xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K (x, x') = 2^{x + x'}$
- $\mathcal{X} = \mathbb{N}, \quad K (x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K (x, x') = \log (1 + xx')$
- $\mathcal{X} = \mathbb{R}, \quad K (x, x') = \exp (-|x - x'|^2)$
- $\mathcal{X} = \mathbb{R}, \quad K (x, x') = \cos (x + x')$
- $\mathcal{X} = \mathbb{R}, \quad K (x, x') = \cos (x - x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K (x, x') = \min(x, x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K (x, x') = \max(x, x')$
Quizz: which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \log(1 + xx')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \exp(-|x - x'|^2)$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos(x + x')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos(x - x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \min(x, x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \max(x, x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \min(x, x')/\max(x, x')$
Quizz : which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \log (1 + xx')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \exp (-|x - x'|^2)$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos (x + x')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \cos (x - x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \min(x, x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \max(x, x')$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \min(x, x')/ \max(x, x')$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = \text{GCD}(x, x')$
Quizz: which of the following are p.d. kernels?

- $\mathcal{X} = (-1, 1)$, $K(x, x') = \frac{1}{1-xx'}$
- $\mathcal{X} = \mathbb{N}$, $K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}$, $K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+$, $K(x, x') = \log(1 + xx')$
- $\mathcal{X} = \mathbb{R}$, $K(x, x') = \exp(-|x-x'|^2)$
- $\mathcal{X} = \mathbb{R}$, $K(x, x') = \cos(x + x')$
- $\mathcal{X} = \mathbb{R}$, $K(x, x') = \cos(x - x')$
- $\mathcal{X} = \mathbb{R}_+$, $K(x, x') = \min(x, x')$
- $\mathcal{X} = \mathbb{R}_+$, $K(x, x') = \max(x, x')$
- $\mathcal{X} = \mathbb{R}_+$, $K(x, x') = \min(x, x')/\max(x, x')$
- $\mathcal{X} = \mathbb{N}$, $K(x, x') = \text{GCD}(x, x')$
- $\mathcal{X} = \mathbb{N}$, $K(x, x') = \text{LCM}(x, x')$
Quizz: which of the following are p.d. kernels?

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

1. Kernels and RKHS
   - Positive Definite Kernels
   - Reproducing Kernel Hilbert Spaces (RKHS)
   - Examples
     - Smoothness functional

2. Kernel tricks

3. Kernel Methods: Supervised Learning


5. The Kernel Jungle

6. Open Problems and Research Topics
Remember the RKHS of the linear kernel

\[
\begin{aligned}
K_{\text{lin}}(x, x') &= x^\top x' \\
f(x) &= w^\top x \\
\|f\|_H &= \|w\|_2
\end{aligned}
\]
A simple inequality

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

$$|f(x) - f(x')| = |\langle f, K_x - K_{x'} \rangle_{\mathcal{H}}|$$

$$\leq \| f \|_{\mathcal{H}} \times \| K_x - K_{x'} \|_{\mathcal{H}}$$

$$= \| f \|_{\mathcal{H}} \times d_K(x, x').$$

- 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 $\Rightarrow$ 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.
Kernel tricks
Motivations

Two theoretical results underpin a family of powerful algorithms for data analysis using p.d. 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.
Motivation from supervised learning

For instance, in supervised learning, the goal is to learn a **prediction function** \( f : \mathcal{X} \to \mathcal{Y} \) given labeled training data \((x_i, y_i)_{i=1,...,n}\) with \( x_i \) in \( \mathcal{X} \), and \( y_i \) in \( \mathcal{Y} \):

\[
\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \Omega(f). 
\]

\( \lambda \Omega(f) \): regularization

\( \mathcal{F} \): function class

\( L \): loss function

\( \lambda \): regularization parameter

\( \Omega(f) \): complexity of the function

(Vapnik, 1995)...
Motivation from supervised learning

For instance, in supervised learning, the goal is to learn a prediction function \( f : \mathcal{X} \to \mathcal{Y} \) given labeled training data \((x_i, y_i)_{i=1,\ldots,n}\) with \(x_i\) in \(\mathcal{X}\), and \(y_i\) in \(\mathcal{Y}\):

\[
\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \Omega(f).
\]

The labels \(y_i\) are, for instance, in

- \([-1, +1]\) for **binary** classification problems.
- \(\{1, \ldots, K\}\) for **multi-class** classification problems.
- \(\mathbb{R}\) for **regression** problems.
- \(\mathbb{R}^k\) for **multivariate regression** problems.
Motivation from supervised learning

For instance, in supervised learning, the goal is to learn a **prediction function** \( f : \mathcal{X} \rightarrow \mathcal{Y} \) given labeled training data \((x_i, y_i)_{i=1,...,n} \) with \( x_i \) in \( \mathcal{X} \), and \( y_i \) in \( \mathcal{Y} \):

\[
\min_{f \in \mathcal{F}} \quad \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \Omega(f) .
\]

**empirical risk, data fit**

**regularization**

Example with linear models: logistic regression, *etc.*

- assume there exists a linear relation between \( y \) and features \( x \) in \( \mathbb{R}^P \).
- \( f(x) = w^\top x + b \) is parametrized by \( w, b \) in \( \mathbb{R}^{p+1} \);
- \( L \) is often a **convex** loss function;
- \( \Omega(f) \) is often the squared \( \ell_2 \)-norm \( \|w\|^2 \).
Motivation from supervised learning

\[
\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \| f \|_{\mathcal{H}}^2.
\]

Kernel methods allow you to map data \( x \) in \( \mathcal{X} \) to a Hilbert space and work with linear forms:

\[
\Phi : \mathcal{X} \rightarrow \mathcal{H} \quad \text{and} \quad f(x) = \langle \Phi(x), f \rangle_{\mathcal{H}}.
\]
Motivation from supervised learning

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

First purpose: embed data in a vectorial space where

- many **geometrical operations** exist (angle computation, projection on linear subspaces, definition of barycenters,...).
- one may learn potentially **rich infinite-dimensional models**.
- **regularization** is natural and theoretically grounded.
Motivation from supervised learning

\[
\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \| f \|^2_{\mathcal{H}}.
\]

First purpose: embed data in a vectorial space where

- many **geometrical operations** exist (angle computation, projection on linear subspaces, definition of barycenters....).
- one may learn potentially **rich infinite-dimensional models**.
- **regularization** is natural and theoretically grounded.

The principle is **generic** and does not assume anything about the nature of the set \( \mathcal{X} \) (vectors, sets, graphs, sequences).
Motivation from supervised learning

Second purpose: unhappy with the current Euclidean structure?

- lift data to a higher-dimensional space with **nicer properties** (e.g., linear separability, clustering structure).
- then, the **linear** form \( f(x) = \langle \Phi(x), f \rangle_\mathcal{H} \) in \( \mathcal{H} \) may correspond to a **non-linear** model in \( \mathcal{X} \).
Outline

1. Kernels and RKHS

2. Kernel tricks
   - The kernel trick
   - The representer theorem

3. Kernel Methods: Supervised Learning


5. The Kernel Jungle

6. Open Problems and Research Topics
The kernel trick

**Proposition**

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.

**Remarks:**

- The proof of this proposition is trivial, because the kernel is exactly the inner product in the feature space.
- This trick has huge practical applications.
- Vectors in the feature space are only manipulated implicitly, through pairwise inner products.
Example 1: computing distances in the feature space

\[ d_K(x_1, x_2)^2 = \| \Phi(x_1) - \Phi(x_2) \|^2_{\mathcal{H}} \]
\[ = \langle \Phi(x_1) - \Phi(x_2), \Phi(x_1) - \Phi(x_2) \rangle_{\mathcal{H}} \]
\[ = \langle \Phi(x_1), \Phi(x_1) \rangle_{\mathcal{H}} + \langle \Phi(x_2), \Phi(x_2) \rangle_{\mathcal{H}} - 2 \langle \Phi(x_1), \Phi(x_2) \rangle_{\mathcal{H}} \]
\[ d_K(x_1, x_2)^2 = K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2) \]
Distance for the Gaussian kernel

- The Gaussian kernel with bandwidth $\sigma$ on $\mathbb{R}^d$ is:

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

- $K(x,x) = 1 = \|\Phi(x)\|^2_{\mathcal{H}}$, so all points are on the unit sphere in the feature space.

- The distance between the images of two points $x$ and $y$ in the feature space is given by:

$$d_K(x,y) = \sqrt{2 \left[ 1 - e^{-\frac{\|x-y\|^2}{2\sigma^2}} \right]}$$
Example 2: distance between a point and a set

Problem

- Let $\mathcal{S} = (x_1, \cdots, x_n)$ be a finite set of points in $\mathcal{X}$.
- How to define and compute the similarity between any point $x$ in $\mathcal{X}$ and the set $\mathcal{S}$?

A solution:

Map all points to the feature space.

Summarize $\mathcal{S}$ by the barycenter of the points: $\mu := \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)$.

Define the distance between $x$ and $\mathcal{S}$ by:

$$d_K(x, \mathcal{S}) := \| \Phi(x) - \mu \|_H.$$
Example 2: distance between a point and a set

Problem

Let $S = (x_1, \cdots, x_n)$ be a finite set of points in $\mathcal{X}$.

How to define and compute the similarity between any point $x$ in $\mathcal{X}$ and the set $S$?

A solution:

Map all points to the feature space.

Summarize $S$ by the barycenter of the points:

$$\mu := \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i).$$

Define the distance between $x$ and $S$ by:

$$d_K(x, S) := \| \Phi(x) - \mu \|_{\mathcal{H}}.$$
Computation

\[ d_K (x, S) = \left\| \Phi(x) - \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i) \right\|_H \]

\[ = \sqrt{K(x, x) - \frac{2}{n} \sum_{i=1}^{n} K(x, x_i) + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} K(x_i, x_j)}. \]

Remark

The barycentre \( \mu \) only exists in the feature space in general: it does not necessarily have a pre-image \( x_\mu \) such that \( \Phi(x_\mu) = \mu \).
\[ S = \{2, 3\} \]

- Plot \( f(x) = d(x, S) \)

\[
K(x, y) = xy.
\]

(Linear)

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

with \( \sigma = 1 \).

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

with \( \sigma = 0.2 \).
1D illustration

- $S = \{2, 3\}$
- Plot $f(x) = d(x, S)$

$$K(x, y) = xy.$$  \hspace{1cm} (linear)  

$$K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}.$$  \hspace{1cm}  with $\sigma = 1.$  

$$K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}.$$  \hspace{1cm}  with $\sigma = 0.2.$

Remarks

- for the linear kernel, $\mathcal{H} = \mathbb{R}$, $\mu = 2.5$ and $d(x, S) = |x - \mu|$.
- for the Gaussian kernel $d(x, S) = \sqrt{C - \frac{2}{n} \sum_{i=1}^{n} K(x_i, x)}$. 
2D illustration

- $S = \{(1, 1)', (1, 2)', (2, 2)\}'$
- Plot $f(x) = d(x, S)$

\[
K(x, y) = xy. \quad (\text{linear})
\]

\[
K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}. \quad \text{with } \sigma = 1.
\]

\[
K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}. \quad \text{with } \sigma = 0.2.
\]
2D illustration

- $S = \{(1, 1)', (1, 2)', (2, 2)\}'$
- Plot $f(x) = d(x, S)$

\[
K(x, y) = xy.
\] (linear)

\[
K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}.
\] with $\sigma = 1.$

\[
K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}.
\] with $\sigma = 0.2.$

**Remark**

- as before, the barycenter $\mu$ in $\mathcal{H}$ (which is a single point in $\mathcal{H}$) may carry a lot of information about the training data.
Basic application in discrimination

- \( S_1 = \{(1, 1)', (1, 2)\}' \) and \( S_2 = \{(1, 3)', (2, 2)\}' \)
- Plot \( f(x) = d(x, S_1)^2 - d(x, S_2)^2 \)

\[
K(x, y) = xy. \quad \text{(linear)}
\]

\[
K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}. \quad \text{with } \sigma = 1.
\]

\[
K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}. \quad \text{with } \sigma = 0.2.
\]
Example 3: Centering data in the feature space

Problem

- Let $S = (x_1, \cdots, x_n)$ be a finite set of points in $\mathcal{X}$ endowed with a p.d. kernel $K$. Let $K$ be their $n \times n$ Gram matrix: $[K]_{ij} = K(x_i, x_j)$.
- Let $\mu = 1/n \sum_{i=1}^{n} \Phi(x_i)$ their barycenter, and $u_i = \Phi(x_i) - \mu$ for $i = 1, \ldots, n$ be centered data in $\mathcal{H}$.
- How to compute the centered Gram matrix $[K^c]_{i,j} = \langle u_i, u_j \rangle_{\mathcal{H}}$?
A direct computation gives, for $0 \leq i, j \leq n$:

\[
K_{i,j}^c = \langle \Phi(x_i) - \mu, \Phi(x_j) - \mu \rangle_H \\
= \langle \Phi(x_i), \Phi(x_j) \rangle_H - \langle \mu, \Phi(x_i) + \Phi(x_j) \rangle_H + \langle \mu, \mu \rangle_H \\
= K_{i,j} - \frac{1}{n} \sum_{k=1}^{n} (K_{i,k} + K_{j,k}) + \frac{1}{n^2} \sum_{k,l=1}^{n} K_{k,l}.
\]

This can be rewritten in matricial form:

\[
K^c = K - UK - KU + UKU = (I - U)K(I - U),
\]

where $U_{i,j} = 1/n$ for $1 \leq i, j \leq n$. 
Kernel trick 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).
Outline

1 Kernels and RKHS

2 Kernel tricks
   - The kernel trick
   - The representer theorem

3 Kernel Methods: Supervised Learning

4 Kernel Methods: Unsupervised Learning

5 The Kernel Jungle

6 Open Problems and Research Topics
Motivation

- An RKHS is a space of (potentially nonlinear) functions, and $\| f \|_H$ measures the smoothness of $f$.

- Given a set of data $(x_i \in \mathcal{X}, y_i \in \mathbb{R})_{i=1,...,n}$, a natural way to estimate a regression function $f : \mathcal{X} \rightarrow \mathbb{R}$ is to solve something like:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda \| f \|_H^2.$$  \hspace{1cm} (1)

for a loss function $\ell$ such as $\ell(y, t) = (y - t)^2$.

- How to solve in practice this problem, potentially in infinite dimension?
The Theorem

Representer Theorem

- Let $\mathcal{X}$ be a set endowed with a p.d. kernel $K$, $\mathcal{H}$ the corresponding RKHS, and $S = \{x_1, \cdots, x_n\} \subseteq \mathcal{X}$ a finite set of points in $\mathcal{X}$.
- Let $\Psi : \mathbb{R}^{n+1} \to \mathbb{R}$ be a function of $n + 1$ variables, strictly increasing with respect to the last variable.
- Then, any solution to the optimization problem:

$$\min_{f \in \mathcal{H}} \Psi(f(x_1), \cdots, f(x_n), \|f\|_{\mathcal{H}}),$$

admits a representation of the form:

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

In other words, the solution lives in a finite-dimensional subspace:

$$f \in \text{Span}(K_{x_1}, \ldots, K_{x_n}).$$
Proof (1/2)

- Let $\xi(f)$ be the functional that is minimized in the statement of the representer theorem, and $\mathcal{H}_S$ the linear span in $\mathcal{H}$ of the vectors $K_{x_i}$:

$$
\mathcal{H}_S = \left\{ f \in \mathcal{H} : f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x), (\alpha_1, \cdots, \alpha_n) \in \mathbb{R}^n \right\}.
$$

- $\mathcal{H}_S$ is a finite-dimensional subspace, therefore any function $f \in \mathcal{H}$ can be uniquely decomposed as:

$$
f = f_S + f_{\perp},
$$

with $f_S \in \mathcal{H}_S$ and $f_{\perp} \perp \mathcal{H}_S$ (by orthogonal projection).
Proof (2/2)

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

\[ \forall i = 1, \ldots, n, \quad f_\perp (x_i) = \langle f_\perp, K_{x_i} \rangle_\mathcal{H} = 0, \]

because $K_{x_i} = K (x_i, .) \in \mathcal{H}_S$ and $f_\perp \perp \mathcal{H}_S$, therefore:

\[ \forall i = 1, \ldots, n, \quad f (x_i) = f_S (x_i). \]

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

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

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

\[ \square \]
Remarks

Often the function $\Psi$ has the form:

$$\Psi (f (x_1), \cdots, f (x_n), \| f \|_H) = c (f (x_1), \cdots, f (x_n)) + \lambda \Omega (\| f \|_H)$$

where $c(.)$ 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 $\| f \|_H$ 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.
Practical use of the representer theorem (1/2)

- When the representer theorem holds, we know that we can look for a solution of the form

\[ f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x), \quad \text{for some } \alpha \in \mathbb{R}^n. \]

- For any \( j = 1, \ldots, n \), we have

\[ f(x_j) = \sum_{i=1}^{n} \alpha_i K(x_i, x_j) = [K\alpha]_j. \]

- Furthermore,

\[ \| f \|_{\mathcal{H}}^2 = \left\| \sum_{i=1}^{n} \alpha_i K_{x_i} \right\|_{\mathcal{H}}^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) = \alpha^\top K \alpha. \]
Therefore, a problem of the form

\[
\min_{f \in \mathcal{H}} \Psi \left( f(x_1), \ldots, f(x_n), \| f \|_{L^2}^2 \right)
\]

is equivalent to the following \( n \)-dimensional optimization problem:

\[
\min_{\alpha \in \mathbb{R}^n} \Psi \left( [K\alpha]_1, \ldots, [K\alpha]_n, \alpha^\top K\alpha \right).
\]

This problem can usually be solved analytically or by numerical methods; we will see many examples in the next sections.
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 $x$ in $\mathcal{X}$, $f(x) = \langle K_x, f \rangle_{\mathcal{H}}$. The part $f^\perp$ that is orthogonal to the $K_{x_i}$’s is thus “useless” to explain the training data.
Kernel Methods
Supervised Learning
Supervised learning

**Definition**

Given:

- $\mathcal{X}$, a space of inputs,
- $\mathcal{Y}$, a space of outputs,
- $S_n = (x_i, y_i)_{i=1,...,n}$, a training set of (input,output) pairs,

the *supervised learning problem* is to estimate a function $h : \mathcal{X} \to \mathcal{Y}$ to predict the output for any future input.

- Regression when $Y = \mathbb{R}$;
- Classification when $Y = \{-1, 1\}$ or any set of two labels;
- Structured output regression or classification when $Y$ is more general.
Supervised learning

Definition

Given:
- $\mathcal{X}$, a space of inputs,
- $\mathcal{Y}$, a space of outputs,
- $S_n = (x_i, y_i)_{i=1,...,n}$, a training set of (input,output) pairs,

the supervised learning problem is to estimate a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict the output for any future input.

Depending on the nature of the output, this covers:
- **Regression** when $\mathcal{Y} = \mathbb{R}$;
- **Classification** when $\mathcal{Y} = \{-1, 1\}$ or any set of two labels;
- **Structured output** regression or classification when $\mathcal{Y}$ is more general.
Example: regression

Task: predict the capacity of a small molecule to inhibit a drug target
\( \mathcal{X} = \text{set of molecular structures (graphs?)} \)
\( \mathcal{Y} = \mathbb{R} \)
Example: classification

Task: recognize if an image is a dog or a cat
\( \mathcal{X} = \) set of images (\( \mathbb{R}^d \))
\( \mathcal{Y} = \{ \text{cat, dog} \} \)
Example: classification

Task: recognize if an image is a dog or a cat
\[ \mathcal{X} = \text{set of images (} \mathbb{R}^d \text{)} \]
\[ \mathcal{Y} = \{ \text{cat, dog} \} \]
Example: structured output

Task: translate from Japanese to French
\( \mathcal{X} = \text{finite-length strings of Japanese characters} \)
\( \mathcal{Y} = \text{finite-length strings of French characters} \)
Supervised learning with kernels: general principles

1. Express $h : \mathcal{X} \rightarrow \mathcal{Y}$ using a real-valued function $f : \mathcal{Z} \rightarrow \mathbb{R}$:
   - regression $\mathcal{Y} = \mathbb{R}$:
     $$h(\mathbf{x}) = f(\mathbf{x}) \quad \text{with} \quad f : \mathcal{X} \rightarrow \mathbb{R} \quad (\mathcal{Z} = \mathcal{X})$$
   - classification $\mathcal{Y} = \{-1, 1\}$:
     $$h(\mathbf{x}) = \text{sign}(f(\mathbf{x})) \quad \text{with} \quad f : \mathcal{X} \rightarrow \mathbb{R} \quad (\mathcal{Z} = \mathcal{X})$$
   - structured output:
     $$h(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} f(\mathbf{x}, y) \quad \text{with} \quad f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R} \quad (\mathcal{Z} = \mathcal{X} \times \mathcal{Y})$$

2. Define an empirical risk function $R_n(f)$ to assess how "good" a candidate function $f$ is on the training set $S_n$, typically the average of a loss:
   $$R_n(f) := \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i)$$

3. Define a p.d. kernel on $\mathcal{Z}$ and solve
   $$\min_{f \in \mathcal{H}, \|f\|_\mathcal{H} \leq B} R_n(f) \quad \text{or} \quad \min_{f \in \mathcal{H}} R_n(f) + \lambda \|f\|_\mathcal{H}^2$$
Remarks

\[
\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) + \lambda \| f \|^2_{\mathcal{H}} \right\}.
\]

- Regularization is important, particularly in high dimension, to prevent **overfitting**
- When \( \mathcal{Z} = \mathbb{R}^d \) and \( K \) is the linear kernel, \( f = f_w \) is a linear model and the regularization is \( \|w\|^2 \)
- Using more general spaces \( \mathcal{Z} \) and kernels \( K \) allows to
  - learn **non-linear functions** over a functional space endowed with a natural regularization (remember, small norm in RKHS = "smooth")
  - learn functions over **non-vectorial data**, such as strings and graphs

We will now see a few methods in more details
Outline

1. Kernels and RKHS

2. Kernel tricks

3. Kernel Methods: Supervised Learning
   - Kernel ridge regression
   - Kernel logistic regression
   - Large-margin classifiers
   - Interlude: convex optimization and duality
   - Support vector machines


5. The Kernel Jungle

6. Open Problems and Research Topics
Regression

Setup

- $\mathcal{X}$ set of inputs
- $\mathcal{Y} = \mathbb{R}$ real-valued outputs
- $S_n = (x_i, y_i)_{i=1,...,n} \in (\mathcal{X} \times \mathbb{R})^n$ a training set of $n$ pairs
- Goal = find a function $f : \mathcal{X} \rightarrow \mathbb{R}$ to predict $y$ by $f(x)$
Regression

Setup

- $\mathcal{X}$ set of inputs
- $\mathcal{Y} = \mathbb{R}$ real-valued outputs
- $S_n = (x_i, y_i)_{i=1,...,n} \in (\mathcal{X} \times \mathbb{R})^n$ a training set of $n$ pairs
- Goal = find a function $f : \mathcal{X} \rightarrow \mathbb{R}$ to predict $y$ by $f(x)$
Least-square regression over a general functional space

- Let us quantify the error if \( f \) predicts \( f(x) \) instead of \( y \) by the squared error:
  \[
  \ell(f(x), y) = (y - f(x))^2
  \]

- Fix a set of functions \( \mathcal{H} \).

- Least-square regression amounts to finding the function in \( \mathcal{H} \) with the smallest empirical risk, called in this case the mean squared error (MSE):
  \[
  \hat{f} \in \arg \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2
  \]

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

Kernel ridge regression (KRR)

- Let us now consider a RKHS $\mathcal{H}$, associated to a p.d. kernel $K$ on $\mathcal{X}$.
- KRR is obtained by regularizing the MSE criterion by the RKHS norm:

$$\hat{f} = \arg \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \| f \|_\mathcal{H}^2$$  \hspace{1cm} (2)

- 1st effect = prevent overfitting by penalizing non-smooth functions.
Kernel ridge regression (KRR)

- Let us now consider a RKHS $\mathcal{H}$, associated to a p.d. kernel $K$ on $\mathcal{X}$.
- KRR is obtained by regularizing the MSE criterion by the RKHS norm:

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

(2)

- 1st effect = prevent overfitting by penalizing non-smooth functions.
- By the representer theorem, any solution of (2) can be expanded as

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

- 2nd effect = simplifying the solution.
Solving KRR

- Let $\mathbf{y} = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$
- Let $\mathbf{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \in \mathbb{R}^n$
- Let $\mathbf{K}$ be the $n \times n$ Gram matrix: $K_{ij} = K(x_i, x_j)$
- We can then write:

$$
\begin{pmatrix}
\hat{f}(x_1), \ldots, \hat{f}(x_n)
\end{pmatrix}^\top = \mathbf{K}\mathbf{\alpha}
$$

- The following holds as usual:

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

- The KRR problem (2) is therefore equivalent to:

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

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

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

\[
0 = \frac{2}{n} K (K\alpha - y) + 2\lambda K\alpha
\]

\[
= K [(K + \lambda n I) \alpha - y]
\]

- For \(\lambda > 0\), \(K + \lambda n I\) is invertible (because \(K\) is positive semidefinite) so one solution is to take:

\[
\alpha = (K + \lambda n I)^{-1} y.
\]
Example (KRR with Gaussian RBF kernel)
Example (KRR with Gaussian RBF kernel)

\[ \lambda = 1000 \]

\[ x \quad y \]

\[ y \quad 0 \quad 1 \quad 2 \]

\[ -1 \quad 0 \quad 1 \quad 2 \]

\[ 0 \quad 2 \quad 4 \quad 6 \quad 8 \quad 10 \]
Example (KRR with Gaussian RBF kernel)

\[ \text{lambda} = 100 \]
Example (KRR with Gaussian RBF kernel)
Example (KRR with Gaussian RBF kernel)
Example (KRR with Gaussian RBF kernel)

\[ \lambda = 0.1 \]
Example (KRR with Gaussian RBF kernel)

\[ \lambda = 0.01 \]
Example (KRR with Gaussian RBF kernel)

\[
\text{lambda} = 0.001
\]
Example (KRR with Gaussian RBF kernel)

\[
\lambda = 0.0001
\]
Example (KRR with Gaussian RBF kernel)

\[ \lambda = 0.00001 \]
Example (KRR with Gaussian RBF kernel)

\[ \lambda = 0.000001 \]
Example (KRR with Gaussian RBF kernel)

\[ \lambda = 0.0000001 \]

![Graph of KRR with Gaussian RBF kernel](image)

- x-axis: \( x \)
- y-axis: \( y \)
Remark: uniqueness of the solution

Let us find all $\alpha$’s that solve

$$K \left[ (K + \lambda n I) \alpha - y \right] = 0$$

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

$$\begin{align*}
(K + \lambda n I) \alpha &- y \in \text{Ker}(K) \\
\iff \alpha &- (K + \lambda n I)^{-1} y \in \text{Ker}(K) \\
\iff \alpha &= (K + \lambda n I)^{-1} y + \epsilon, \text{ with } K \epsilon = 0.
\end{align*}$$

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

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

therefore $f = f'$. KRR has a unique solution $f \in \mathcal{H}$, which can possibly be expressed by several $\alpha$’s if $K$ is singular.
Remark: link with "standard" ridge regression

- Take $\mathcal{X} = \mathbb{R}^d$ and the linear kernel $K(x, x') = x^\top x'$
- Let $X = (x_1, \ldots, x_n)^\top$ the $n \times d$ data matrix
- The kernel matrix is then $K = XX^\top$
- The function learned by KRR in that case is linear:

$$f_{KRR}(x) = w_{KRR}^\top x$$

with

$$w_{KRR} = \sum_{i=1}^n \alpha_i x_i = X^\top \alpha = X^\top \left(XX^\top + \lambda n I\right)^{-1} y$$
Remark: link with ”standard” ridge regression

- On the other hand, the RKHS is the set of linear functions $f(x) = w^T x$ and the RKHS norm is $\| f \|_{\mathcal{H}} = \| w \|
- We can therefore directly rewrite the original KRR problem (2) as

$$\arg \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left( y_i - w^T x_i \right)^2 + \lambda \| w \|^2$$

$$= \arg \min_{w \in \mathbb{R}^d} \frac{1}{n} (y - Xw)^T (y - Xw) + \lambda w^T w$$

- Setting the gradient to 0 gives the solution:

$$w_{RR} = \left( X^T X + \lambda n I \right)^{-1} X^T y$$

- Oups, looks different from $w_{KRR} = X^T (XX^T + \lambda n I)^{-1} y$ ..?
Remark: link with "standard" ridge regression

Matrix inversion lemma

For any matrices $B$ and $C$, and $\gamma > 0$ the following holds (when it makes sense):

$$B (CB + \gamma I)^{-1} = (BC + \gamma I)^{-1} B$$

We deduce that (of course...):

$$w_{RR} = \left( X^T X + \lambda n I \right)^{-1} X^T y = X^T \left( X X^T + \lambda n I \right)^{-1} y = w_{KRR}$$

Computationally, inverting the matrix is the expensive part, which suggests to implement:

KRR when $d > n$ (high dimension)

RR when $d < n$ (many points)
Remark: link with “standard” ridge regression

Matrix inversion lemma

For any matrices $B$ and $C$, and $\gamma > 0$ the following holds (when it makes sense):

$$B (CB + \gamma I)^{-1} = (BC + \gamma I)^{-1} B$$

We deduce that (of course...):

$$w_{RR} = \left(X^T X + \lambda n I\right)^{-1} X^T y = X^T \left(X X^T + \lambda n I\right)^{-1} y = w_{KRR}$$

Computationally, inverting the matrix is the expensive part, which suggest to implement:

- KRR when $d > n$ (high dimension)
- RR when $d < n$ (many points)
Robust regression

- The squared error $\ell(t, y) = (t - y)^2$ is arbitrary and sensitive to outliers.
- Many other loss functions exist for regression, e.g.:

Any loss function leads to a valid kernel method, which is usually solved by numerical optimization as there is usually no analytical solution beyond the squared error.
Weighted regression

Given weights $W_1, \ldots, W_n \in \mathbb{R}$, a variant of ridge regression is to weight differently the error at different points:

$$\arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} W_i (y_i - f(x_i))^2 + \lambda \|f\|^2_\mathcal{H}$$

By the representer theorem the solution is $f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)$ where $\alpha$ solves, with $W = \text{diag}(W_1, \ldots, W_n)$:

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

Setting the gradient to zero gives

\[ 0 = \frac{2}{n} (KWK\alpha - KWy) + 2\lambda K\alpha \]

\[ = \frac{2}{n} KW^{1/2} \left[ \left( W^{1/2}KW^{1/2} + n\lambda I \right) W^{-1/2} \alpha - W^{1/2}y \right] \]

A solution is therefore given by

\[ \left( W^{1/2}KW^{1/2} + n\lambda I \right) W^{-1/2} \alpha - W^{1/2}y = 0 \]

therefore

\[ \alpha = W^{1/2} \left( W^{1/2}KW^{1/2} + n\lambda I \right)^{-1} W^{1/2}y \]
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
   - Kernel ridge regression
   - Kernel logistic regression
   - Large-margin classifiers
   - Interlude: convex optimization and duality
   - Support vector machines
5. The Kernel Jungle
6. Open Problems and Research Topics
Binary classification

Setup

- $\mathcal{X}$ set of inputs
- $\mathcal{Y} = \{-1, 1\}$ binary outputs
- $S_n = (x_i, y_i)_{i=1,\ldots,n} \in (\mathcal{X} \times \mathcal{Y})^n$ a training set of $n$ pairs
- Goal = find a function $f : \mathcal{X} \to \mathbb{R}$ to predict $y$ by $\text{sign}(f(x))$
Binary classification

Setup

- $\mathcal{X}$ set of inputs
- $\mathcal{Y} = \{-1, 1\}$ binary outputs
- $S_n = (x_i, y_i)_{i=1,...,n} \in (\mathcal{X} \times \mathcal{Y})^n$ a training set of $n$ pairs
- Goal: find a function $f : \mathcal{X} \rightarrow \mathbb{R}$ to predict $y$ by $\text{sign}(f(x))$
The 0/1 loss

- The 0/1 loss measures if a prediction is correct or not:

\[
\ell_{0/1}(f(x), y) = 1 \text{(} yf(x) < 0 \text{)} = \begin{cases} 
0 & \text{if } y = \text{sign}(f(x)) \\
1 & \text{otherwise.}
\end{cases}
\]

- It is then tempting to learn \( f \) by solving:

\[
\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell_{0/1}(f(x_i), y_i) + \lambda \| f \|_{\mathcal{H}}^2
\]

  \underbrace{\text{misclassification rate}}_{\text{regularization}}

- However:
  - The problem is non-smooth, and typically NP-hard to solve
  - The regularization has no effect since the 0/1 loss is invariant by scaling of \( f \)
  - In fact, no function achieves the minimum when \( \lambda > 0 \) (why?)
The logistic loss

- An alternative is to define a probabilistic model of $y$ parametrized by $f(x)$, e.g.:

$$\forall y \in \{-1, 1\}, \quad p(y \mid f(x)) = \frac{1}{1 + e^{-yf(x)}} = \sigma(yf(x))$$

- The **logistic loss** is the negative conditional likelihood:

$$\ell_{\text{logistic}}(f(x), y) = -\ln p(y \mid f(x)) = \ln \left(1 + e^{-yf(x)}\right)$$
Kernel logistic regression (KLR)

\[ \hat{f} = \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell_{\text{logistic}}(f(x_i), y_i) + \frac{\lambda}{2} \| f \|_{\mathcal{H}}^{2} \]

\[ = \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ln \left( 1 + e^{-y_i f(x_i)} \right) + \frac{\lambda}{2} \| f \|_{\mathcal{H}}^{2} \]

- Can be interpreted as a regularized conditional maximum likelihood estimator
- No explicit solution, but smooth convex optimization problem that can be solved numerically
Solving KLR

- By the representer theorem, any solution of KLR can be expanded as

\[ \hat{f}(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) \]

and as always we have:

\[ \left( \hat{f}(x_1), \ldots, \hat{f}(x_n) \right)^\top = K\alpha \quad \text{and} \quad \| \hat{f} \|_H^2 = \alpha^\top K\alpha \]

- To find \( \alpha \) we therefore need to solve:

\[
\min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \ln \left( 1 + e^{-y_i [K\alpha]_i} \right) + \frac{\lambda}{2} \alpha^\top K\alpha
\]
Technical facts

Sigmoid:
- $\sigma(u) = \frac{1}{1 + e^{-u}}$
- $\sigma(-u) = 1 - \sigma(u)$
- $\sigma'(u) = \sigma(u)\sigma(-u) \geq 0$

Logistic loss:
- $\ell_{logistic}(u) = \ln(1 + e^{-u})$
- $\ell'_{logistic}(u) = -\sigma(-u)$
- $\ell''_{logistic}(u) = \sigma(u)\sigma(-u) \geq 0$
This is a smooth convex optimization problem, that can be solved by many numerical methods. Let us explicit one of them, Newton’s method, which iteratively approximates $J$ by a quadratic function and solves the quadratic problem.

The quadratic approximation near a point $\alpha_0$ is the function:

$$J_q(\alpha) = J(\alpha_0) + (\alpha - \alpha_0)^\top \nabla J(\alpha_0) + \frac{1}{2} (\alpha - \alpha_0)^\top \nabla^2 J(\alpha_0) (\alpha - \alpha_0)$$

Let us compute the different terms...
Computing the quadratic approximation

\[
\frac{\partial J}{\partial \alpha_j} = \frac{1}{n} \sum_{i=1}^{n} \ell'_{\text{logistic}}(y_i [K\alpha]_i) y_i K_{ij} + \lambda [K\alpha]_j
\]

therefore

\[
\nabla J(\alpha) = \frac{1}{n} K P(\alpha) y + \lambda K \alpha
\]

where \( P(\alpha) = \text{diag}(P_1(\alpha), \ldots, P_n(\alpha)) \).

\[
\frac{\partial^2 J}{\partial \alpha_j \partial \alpha_l} = \frac{1}{n} \sum_{i=1}^{n} \ell''_{\text{logistic}}(y_i [K\alpha]_i) y_i K_{ij} y_i K_{il} + \lambda K_{jl}
\]

therefore

\[
\nabla^2 J(\alpha) = \frac{1}{n} K W(\alpha) K + \lambda K
\]

where \( W(\alpha) = \text{diag}(W_1(\alpha), \ldots, W_n(\alpha)) \).
Computing the quadratic approximation

\[ J_q(\alpha) = J(\alpha_0) + (\alpha - \alpha_0)^\top \nabla J(\alpha_0) + \frac{1}{2} (\alpha - \alpha_0)^\top \nabla^2 J(\alpha_0) (\alpha - \alpha_0) \]

Terms that depend on \( \alpha \), with \( P = P(\alpha_0) \) and \( W = W(\alpha_0) \):

- \( \alpha^\top \nabla J(\alpha_0) = \frac{1}{n} \alpha^\top KPy + \lambda \alpha^\top Ka_0 \)
- \( \frac{1}{2} \alpha^\top \nabla^2 J(\alpha_0) \alpha = \frac{1}{2n} \alpha^\top KWK\alpha + \frac{1}{2} \alpha^\top K\alpha \)
- \( -\alpha^\top \nabla^2 J(\alpha_0) \alpha_0 = -\frac{1}{n} \alpha^\top KWK\alpha_0 - \lambda \alpha^\top Ka_0 \)

Putting it all together:

\[ 2J_q(\alpha) = -\frac{2}{n} \alpha^\top KW (K\alpha_0 - W^{-1}Py) + \frac{1}{n} \alpha^\top KWK\alpha + \lambda \alpha^\top K\alpha + C \]

\[ = \frac{1}{n} (K\alpha - z)^\top W (K\alpha - z) + \lambda \alpha^\top K\alpha + C \]

This is a standard weighted kernel ridge regression (WKRR) problem!
Solving KLR by IRLS

In summary, one way to solve KLR is to iteratively solve a WKRR problem until convergence:

\[ \alpha^{t+1} \leftarrow \text{solveWKRR}(K, W^t, z^t) \]

where we update \( W^t \) and \( z^t \) from \( \alpha^t \) as follows (for \( i = 1, \ldots, n \)):

- \( m_i \leftarrow [K\alpha^t]_i \)
- \( P_i^t \leftarrow \ell'_\text{logistic}(y_i m_i) = -\sigma(-y_i m_i) \)
- \( W_i^t \leftarrow \ell''_\text{logistic}(y_i m_i) = \sigma(m_i)\sigma(-m_i) \)
- \( z_i^t \leftarrow m_i - P_i^t y_i / W_i^t = m_i + y_i / \sigma(y_i m_i) \)

This is the kernelized version of the famous iteratively reweighted least-square (IRLS) method to solve the standard linear logistic regression.
Outline

1. Kernels and RKHS

2. Kernel tricks

3. Kernel Methods: Supervised Learning
   - Kernel ridge regression
   - Kernel logistic regression
   - Large-margin classifiers
   - Interlude: convex optimization and duality
   - Support vector machines


5. The Kernel Jungle

6. Open Problems and Research Topics
Loss functions for classifications

We already saw two loss functions for binary classification problems

- The 0/1 loss \( \ell_{0/1}(f(x), y) = 1(yf(x) < 0) \)
- The logistic loss \( \ell_{\text{logistic}}(f(x), y) = \ln(1 + e^{-yf(x)}) \)

In both cases, the loss is a function of the margin defined as follows

**Definition**

In binary classification \( (Y = \{-1, 1\}) \), the **margin** of the function \( f \) for a pair \( (x, y) \) is:

\[
yf(x).
\]

In both cases the loss is a decreasing function of the margin, i.e.,

\[
\ell(f(x), y) = \varphi(yf(x)), \quad \text{with } \varphi \text{ non-increasing}
\]

What about other similar loss functions?
## Loss function examples

<table>
<thead>
<tr>
<th>Method</th>
<th>$\phi(u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel logistic regression</td>
<td>$\log(1 + e^{-u})$</td>
</tr>
<tr>
<td>Support vector machine (1-SVM)</td>
<td>$\max(1 - u, 0)$</td>
</tr>
<tr>
<td>Support vector machine (2-SVM)</td>
<td>$\max(1 - u, 0)^2$</td>
</tr>
<tr>
<td>Boosting</td>
<td>$e^{-u}$</td>
</tr>
</tbody>
</table>
Large-margin classifiers

**Definition**

Given a non-increasing function $\varphi : \mathbb{R} \rightarrow \mathbb{R}_+$, a (kernel) large-margin classifier is an algorithm that estimates a function $f : \mathcal{X} \rightarrow \mathbb{R}$ by solving

$$
\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \varphi (y_i f(x_i)) + \lambda \| f \|_{L^2_{\mathcal{H}}}
$$

Hence, KLR is a large-margin classifier, corresponding to $\varphi(u) = \ln (1 + e^{-u})$. Many more are possible.

**Questions:**

1. Can we solve the optimization problem for other $\varphi$’s?
2. Is it a good idea to optimize this objective function, if at the end of the day we are interested in the $\ell_{0/1}$ loss, i.e., learning models that make few errors?
Solving large-margin classifiers

\[ \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \varphi (y_i f(x_i)) + \lambda \| f \|_{\mathcal{H}}^2 \]

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

\[ f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x). \]

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

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

- When \( \varphi \) is convex, this can be solved using general tools for convex optimization, or specific algorithms (e.g., for SVM, see later).
A tiny bit of learning theory

Assumptions and notations

- Let $\mathbb{P}$ be an (unknown) distribution on $\mathcal{X} \times \mathcal{Y}$, and $\eta(x) = \mathbb{P}(Y = 1 \mid X = x)$ a measurable version of the conditional distribution of $Y$ given $X$.
- Assume the training set $S_n = (X_i, Y_i)_{i=1,\ldots,n}$ are i.i.d. random variables according to $\mathbb{P}$.
- The risk of a classifier $f : \mathcal{X} \to \mathbb{R}$ is $R(f) = \mathbb{P}(\text{sign}(f(X)) \neq Y)$.
- The Bayes risk is
  \[ R^* = \inf_{f \text{ measurable}} R(f) \]
  which is attained for $f^*(x) = \eta(x) - 1/2$.
- The empirical risk of a classifier $f : \mathcal{X} \to \mathbb{R}$ is
  \[ R^n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(\text{sign}(f(X_i)) \neq Y_i) \]
Let the empirical $\varphi$-risk be the empirical risk optimized by a large-margin classifier:

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

It is the empirical version of the $\varphi$-risk

$$R_\varphi(f) = \mathbb{E}[\varphi(Y f(X))]$$

Can we hope to have a small risk $R(f)$ if we focus instead on the $\varphi$-risk $R_\varphi(f)$?
A small $\varphi$-risk ensures a small 0/1 risk

**Theorem (Bartlett et al., 2003)**

Let $\varphi : \mathbb{R} \to \mathbb{R}_+$ be convex, non-increasing, differentiable at 0 with $\varphi'(0) < 0$. Let $f : \mathcal{X} \to \mathbb{R}$ measurable such that

$$R_{\varphi}(f) = \min_{g \text{ measurable}} R_{\varphi}(g) = R^*_\varphi.$$ 

Then

$$R(f) = \min_{g \text{ measurable}} R(g) = R^*.$$ 

**Remarks:**

- This tells us that, if we know $\mathbb{P}$, then minimizing the $\varphi$-risk is a good idea even if our focus is on the classification error.
- The assumptions on $\varphi$ can be relaxed; it works for the broader class of *classification-calibrated* loss functions (Bartlett et al., 2003).
- More generally, we can show that if $R_{\varphi}(f) - R^*_\varphi$ is small, then $R(f) - R^*$ is small too (Bartlett et al., 2003).
A small $\varphi$-risk ensures a small 0/1 risk

Proof sketch: Show that $f(x)$ is necessarily consistent with $\eta(x) = \mathbb{P}(Y = 1 \mid X = x)$, if $f$ minimizes $R_{\varphi}$, and thus minimizes $R$.

Condition on $X = x$:

$$R_{\varphi}(f \mid X = x) = \mathbb{E}[\varphi(Yf(X)) \mid X = x] = \eta(x)\varphi(f(x)) + (1 - \eta(x))\varphi(-f(x))$$
$$R_{\varphi}(-f \mid X = x) = \mathbb{E}[\varphi(-Yf(X)) \mid X = x] = \eta(x)\varphi(-f(x)) + (1 - \eta(x))\varphi(f(x))$$

Therefore:

$$R_{\varphi}(f \mid X = x) - R_{\varphi}(-f \mid X = x) = [2\eta(x) - 1] \times [\varphi(f(x)) - \varphi(-f(x))]$$

This must be a.s. $\leq 0$ because $R_{\varphi}(f) \leq R_{\varphi}(-f)$, which implies:

- if $\eta(x) > \frac{1}{2}$, $\varphi(f(x)) \leq \varphi(-f(x)) \implies f(x) \geq 0$
- if $\eta(x) < \frac{1}{2}$, $\varphi(f(x)) \geq \varphi(-f(x)) \implies f(x) \leq 0$

These inequalities are in fact strict thanks to the assumptions we made on $\varphi$ (*left as exercise*).
Empirical risk minimization (ERM)

To find a function with a small $\varphi$-risk, the following is a good candidate:

**Definition**

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

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

**Questions**

Is $R^n_\varphi(\hat{f}_n)$ a good estimate of the true risk $R_\varphi(\hat{f}_n)$?

Is $R_\varphi(\hat{f}_n)$ small?

$$R_\varphi(\hat{f}_n) - R_\varphi^* = R_\varphi(\hat{f}_n) - \inf_{f \in \mathcal{F}} R_\varphi(f) = \text{estimation error} + \inf_{f \in \mathcal{F}} R_\varphi(f) - R_\varphi^* = \text{approximation error}.$$
Empirical risk minimization (ERM)

To find a function with a small $\varphi$-risk, the following is a good candidate:

**Definition**

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

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

**Questions**

- Is $R^n_\varphi(f)$ a good estimate of the true risk $R_\varphi(f)$?
- Is $R_\varphi(\hat{f}_n)$ small?
Empirical risk minimization (ERM)

To find a function with a small $\varphi$-risk, the following is a good candidate:

**Definition**

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

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

**Questions**

- Is $R_n^\varphi(f)$ a good estimate of the true risk $R_\varphi(f)$?
- Is $R_\varphi(\hat{f}_n)$ small?

$$R_\varphi(\hat{f}_n) - R^*_\varphi = R_\varphi(\hat{f}_n) - \inf_{f \in \mathcal{F}} R_\varphi(f) + \inf_{f \in \mathcal{F}} R_\varphi(f) - R^*_\varphi.$$

- **estimation error**
- **approximation error**
Class capacity

Motivations

- The ERM principle gives a good solution if $R_\varphi \left( \hat{f}_n \right)$ 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 $(X_i)_{i=1,\ldots,n}$ and the independent uniform $\{\pm 1\}$-valued (Rademacher) random variables $(\sigma_i)_{i=1,\ldots,n}$. 
Basic learning bounds

**Theorem**
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 the $\varphi$-risk of the ERM estimator satisfies (on average over the sampling of training set)

$$\mathbb{E} \mathcal{S}_n R_\varphi \left( \hat{f}_n \right) - R^*_\varphi \leq 4L_\varphi \text{Rad}_n(\mathcal{F}) + \inf_{f \in \mathcal{F}} R_\varphi(f) - R^*_\varphi$$

This quantifies a trade-off between:

- $\mathcal{F}$ "large" = **overfitting** (approximation error small, estimation error large)
- $\mathcal{F}$ "small" = **underfitting** (estimation error small, approximation error large)
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:
  \[ F_B = \{ f \in \mathcal{H} : \| f \|_{\mathcal{H}} \leq B \} . \]

Theorem (capacity control of RKHS balls)

\[ \text{Rad}_n (F_B) \leq \frac{2B \sqrt{\mathbb{E} K(X, X)}}{\sqrt{n}} . \]
Proof (1/2)

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

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

$$= \mathbb{E}_{X, \sigma} \left[ B \left\| \frac{2}{n} \sum_{i=1}^{n} \sigma_i K_{X_i} \right\|_{\mathcal{H}} \right] \quad \text{(Cauchy-Schwarz)}$$

$$= \frac{2B}{n} \mathbb{E}_{X, \sigma} \left[ \left( \sum_{i=1}^{n} \sigma_i K_{X_i} \right)^{\frac{2}{2}} \right]$$

$$\leq \frac{2B}{n} \mathbb{E}_{X, \sigma} \left[ \sum_{i, j=1}^{n} \sigma_i \sigma_j K (X_i, X_j) \right] \quad \text{(Jensen)}$$
Proof (2/2)

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

$$\text{Rad}_n (\mathcal{F}_B) \leq \frac{2B}{n} \sqrt{\mathbb{E}_X \left[ \sum_{i,j=1}^{n} \mathbb{E}_\sigma [\sigma; \sigma_j] K(X_i, X_j) \right]}$$

$$\leq \frac{2B}{n} \sqrt{\mathbb{E}_X \sum_{i=1}^{n} K(X_i, X_i)}$$

$$= \frac{2B \sqrt{\mathbb{E}_X K(X, X)}}{\sqrt{n}}. \quad \square$$
Basic learning bounds in RKHS balls

**Corollary**

Suppose $K(X, X) \leq \kappa^2$ a.s. (e.g., Gaussian kernel and $\kappa = 1$). Then the ERM estimator in $\mathcal{F}_B$ satisfies

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

**Remarks**

- $B$ controls the trade-off between approximation and estimation error.
- The bound on expression error is independent of $\mathcal{P}$ and decreases with $n$.
- The approximation error is harder to analyze in general.
- In practice, $B$ (or $\lambda$, next slide) is tuned by cross-validation.
ERM as penalized risk minimization

- ERM over $\mathcal{F}_B$ solves the constrained minimization problem:

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

- 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:

$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \varphi(y_i f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2 \right\}.$$
Summary: large margin classifiers

\[
\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \varphi(y_i f(x_i)) + \lambda \| f \|_2^2 \right\}
\]

- \( \varphi \) calibrated (e.g., decreasing, \( \varphi'(0) < 0 \)) \( \Rightarrow \) good proxy for classification error
- \( \varphi \) convex + representer theorem \( \Rightarrow \) efficient algorithms
Outline

1 Kernels and RKHS

2 Kernel tricks

3 Kernel Methods: Supervised Learning
   - Kernel ridge regression
   - Kernel logistic regression
   - Large-margin classifiers
   - Interlude: convex optimization and duality
   - Support vector machines

4 Kernel Methods: Unsupervised Learning

5 The Kernel Jungle

6 Open Problems and Research Topics
Strong duality means that $\max_{\nu} q(\nu) = \min_x f(x)$

Strong duality holds in most “reasonable cases” for convex optimization (to be detailed soon).
A few slides on convex duality

Strong Duality

The relation between $x^*$ and $\nu^*$ is not always known a priori.
The duality gap guarantees us that $0 \leq f(\tilde{x}) - f(x^*) \leq \delta(\tilde{x}, \tilde{\nu})$.

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 \( x \in \mathcal{X} \):

  \[
  \begin{align*}
  &\text{minimize} & f(x) \\
  &\text{subject to} & h_i(x) = 0 , \quad i = 1, \ldots, m , \\
  & & g_j(x) \leq 0 , \quad j = 1, \ldots, r ,
  \end{align*}
  \]

  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:

$$q(\lambda, \mu) = \inf_{x \in \mathcal{X}} L(x, \lambda, \mu)$$

$$= \inf_{x \in \mathcal{X}} \left( f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{r} \mu_j g_j(x) \right).$$
A few slides on convex Lagrangian duality

For the (primal) problem:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad h(x) = 0, \quad g(x) \leq 0,
\end{align*}
\]

the Lagrange dual problem is:

\[
\begin{align*}
\text{maximize} & \quad q(\lambda, \mu) \\
\text{subject to} & \quad \mu \geq 0,
\end{align*}
\]

Proposition

- \( q \) is concave in \((\lambda, \mu)\), even if the original problem is not convex.
- The dual function yields lower bounds on the optimal value \( f^* \) of the original problem when \( \mu \) is nonnegative:

\[
q(\lambda, \mu) \leq f^*, \quad \forall \lambda \in \mathbb{R}^m, \forall \mu \in \mathbb{R}^r, \mu \geq 0.
\]
Proofs

- Remember that

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

- For each \( x \), the function \((\lambda, \mu) \mapsto L(x, \lambda, \mu)\) is linear, and therefore both convex and concave in \((\lambda, \mu)\). The pointwise minimum of concave functions is concave, therefore \( q \) is concave.

- Let \( \bar{x} \) be any feasible point, i.e., \( h(\bar{x}) = 0 \) and \( g(\bar{x}) \leq 0 \). Then we have, for any \( \lambda \) and \( \mu \geq 0 \):

\[
\sum_{i=1}^{m} \lambda_i h_i(\bar{x}) + \sum_{i=1}^{r} \mu_i g_i(\bar{x}) \leq 0,
\]

\[
\implies L(\bar{x}, \lambda, \mu) = f(\bar{x}) + \sum_{i=1}^{m} \lambda_i h_i(\bar{x}) + \sum_{i=1}^{r} \mu_i g_i(\bar{x}) \leq f(\bar{x}),
\]

\[\implies q(\lambda, \mu) = \inf_{x} L(x, \lambda, \mu) \leq L(\bar{x}, \lambda, \mu) \leq f(\bar{x}), \quad \forall \bar{x}. \]
Let $q^*$ the optimal value of the Lagrange dual problem. Each $q(\lambda, \mu)$ is a lower bound for $f^*$ and by definition $q^*$ is the best lower bound that is obtained. The following weak duality inequality therefore always hold:

$$q^* \leq f^*.$$ 

This inequality holds when $q^*$ or $f^*$ are infinite. The difference $q^* - f^*$ is called the optimal duality gap of the original problem.
We say that strong duality holds if the optimal duality gap is zero, i.e.:

\[ q^* = f^* . \]

If strong duality holds, then the best lower bound that can be obtained from the Lagrange dual function is tight.

Strong duality does not hold for general nonlinear problems.

It usually holds for convex problems.

Conditions that ensure strong duality for convex problems are called constraint qualification.

In that case, we have for all feasible primal and dual points \( x, \lambda, \mu, \)

\[ q(\lambda, \mu) \leq q(\lambda^*, \mu^*) = L(x^*, \lambda^*, \mu^*) = f(x^*) \leq f(x). \]
Slater’s constraint qualification

Strong duality holds for a convex problem:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g_j(x) \leq 0, \quad j = 1, \ldots, r, \\
& \quad Ax = b,
\end{align*}
\]

if it is strictly feasible, i.e., there exists at least one feasible point that satisfies:

\[
\begin{align*}
g_j(x) & < 0, \quad j = 1, \ldots, r, \quad Ax = b.
\end{align*}
\]
Remarks

- Slater’s conditions also ensure that the maximum $q^*$ (if $> -\infty$) is attained, i.e., there exists a point $(\lambda^*, \mu^*)$ with
  
  $$q(\lambda^*, \mu^*) = q^* = f^*$$

- They can be sharpened. For example, strict feasibility is not required for affine constraints.

- There exist many other types of constraint qualifications
Dual optimal pairs

Suppose that strong duality holds, $x^*$ is primal optimal, $(\lambda^*, \mu^*)$ is dual optimal. Then we have:

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

$$= \inf_{x \in \mathbb{R}^n} \left\{ f(x) + \sum_{i=1}^{m} \lambda_i^* h_i(x) + \sum_{j=1}^{r} \mu_j^* g_j(x) \right\}$$

$$\leq f(x^*) + \sum_{i=1}^{m} \lambda_i^* h_i(x^*) + \sum_{j=1}^{r} \mu_j^* g_j(x^*)$$

$$\leq f(x^*)$$

Hence both inequalities are in fact equalities.
Complimentary slackness

The first equality shows that:

\[ L(x^*, \lambda^*, \mu^*) = \inf_{x \in \mathbb{R}^n} L(x, \lambda^*, \mu^*) , \]

showing that \( x^* \) minimizes the Lagrangian at \( (\lambda^*, \mu^*) \). The second equality shows the following important property:

Complimentary slackness

Each optimal Lagrange multiplier is zero unless the corresponding constraint is active at the optimum:

\[ \mu_j g_j(x^*) = 0 , \quad j = 1, \ldots, r . \]
Outline

1 Kernels and RKHS

2 Kernel tricks

3 Kernel Methods: Supervised Learning
   • Kernel ridge regression
   • Kernel logistic regression
   • Large-margin classifiers
   • Interlude: convex optimization and duality
   • Support vector machines

4 Kernel Methods: Unsupervised Learning

5 The Kernel Jungle

6 Open Problems and Research Topics
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).
Support vector machines (SVM)

**Definition**

- **The hinge loss** is the function \( \mathbb{R} \to \mathbb{R}_+ \):

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

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

\[
\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \varphi_{\text{hinge}}(y_i f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2 \right\}.
\]
Problem reformulation (1/3)

- By the representer theorem, the solution satisfies

\[
\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x_i, x),
\]

where \(\hat{\alpha}\) solves

\[
\min_{\alpha \in \mathbb{R}^n} \left\{ \frac{1}{n} \sum_{i=1}^{n} \varphi_{\text{hinge}}(y_i [K\alpha]_i) + \lambda \alpha^\top K\alpha \right\}
\]

- This is a convex optimization problem
- But the objective function is not smooth (because of the hinge loss)
Let us introduce additional slack variables $\xi_1, \ldots, \xi_n \in \mathbb{R}$. The problem is equivalent to:

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

subject to:

$$\xi_i \geq \varphi_{\text{hinge}}(y_i[K\alpha]_i).$$

The objective function is now smooth, but not the constraints. However it is easy to replace the non-smooth constraint by a conjunction of two smooth constraints, because:

$$u \geq \varphi_{\text{hinge}}(v) \iff \begin{cases} u \geq 1 - v \\ u \geq 0 \end{cases}$$
Problem reformulation (3/3)

In summary, the SVM solution is

\[
\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x_i, x),
\]

where \( \hat{\alpha} \) solves:

**SVM (primal formulation)**

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

subject to:

\[
\begin{cases}
    y_i [K\alpha]_i + \xi_i - 1 \geq 0, & \text{for } i = 1, \ldots, n, \\
    \xi_i \geq 0, & \text{for } i = 1, \ldots, n.
\end{cases}
\]
Solving the SVM problem

- 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).
Let us introduce the Lagrange multipliers $\mu \in \mathbb{R}^n$ and $\nu \in \mathbb{R}^n$.

The Lagrangian of the problem is:

$$L(\alpha, \xi, \mu, \nu) = \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \alpha^\top K \alpha$$

$$- \sum_{i=1}^{n} \mu_i [y_i[K\alpha]_i + \xi_i - 1] - \sum_{i=1}^{n} \nu_i \xi_i$$

or, in matrix notations:

$$L(\alpha, \xi, \mu, \nu) = \xi^\top \frac{1}{n} + \lambda \alpha^\top K \alpha$$

$$- (\text{diag}(y)\mu)^\top K \alpha - (\mu + \nu)^\top \xi + \mu^\top 1$$
Minimizing $L(\alpha, \xi, \mu, \nu)$ w.r.t. $\alpha$

- $L(\alpha, \xi, \mu, \nu)$ is a convex quadratic function in $\alpha$. It is minimized whenever its gradient is null:

$$\nabla_\alpha L = 2\lambda K \alpha - K \text{diag}(y) \mu = K (2\lambda \alpha - \text{diag}(y) \mu)$$

- The following solves $\nabla_\alpha L = 0$:

$$\alpha^* = \frac{\text{diag}(y) \mu}{2\lambda}$$
Minimizing $L(\alpha, \xi, \mu, \nu)$ w.r.t. $\xi$

- $L(\alpha, \xi, \mu, \nu)$ is a linear function in $\xi$.
- Its minimum is $-\infty$ except when it is constant, i.e., when:
  \[
  \nabla_\xi L = \frac{1}{n} - \mu - \nu = 0
  \]
  or equivalently
  \[
  \mu + \nu = \frac{1}{n}
  \]
Dual function

- We therefore obtain the Lagrange dual function:

\[
q(\mu, \nu) = \inf_{\alpha \in \mathbb{R}^n, \xi \in \mathbb{R}^n} L(\alpha, \xi, \mu, \nu)
\]

\[
= \begin{cases} 
\mu^\top 1 - \frac{1}{4\lambda} \mu^\top \text{diag}(y)K \text{diag}(y) \mu & \text{if } \mu + \nu = \frac{1}{n}, \\
-\infty & \text{otherwise.}
\end{cases}
\]

- The dual problem is:

\[
\begin{align*}
\text{maximize} & \quad q(\mu, \nu) \\
\text{subject to} & \quad \mu \geq 0, \nu \geq 0.
\end{align*}
\]
Dual problem

- If $\mu_i > 1/n$ for some $i$, then there is no $\nu_i \geq 0$ such that $\mu_i + \nu_i = 1/n$, hence $q(\mu, \nu) = -\infty$.
- If $0 \leq \mu_i \leq 1/n$ for all $i$, then the dual function takes finite values that depend only on $\mu$ by taking $\nu_i = 1/n - \mu_i$.
- The dual problem is therefore equivalent to:

$$\max_{0 \leq \mu \leq 1/n} \mu^\top 1 - \frac{1}{4\lambda} \mu^\top \text{diag}(y)K \text{diag}(y) \mu$$

or with indices:

$$\max_{0 \leq \mu \leq 1/n} \sum_{i=1}^{n} \mu_i - \frac{1}{4\lambda} \sum_{i,j=1}^{n} y_i y_j \mu_i \mu_j K(x_i, x_j).$$
Back to the primal

Once the dual problem is solved in $\mu$ we get a solution of the primal problem by $\alpha = \text{diag}(y)\mu/2\lambda$.

Because the link is so simple, we can therefore directly plug this into the dual problem to obtain the QP that $\alpha$ must solve:

**SVM (dual formulation)**

$$\max_{\alpha \in \mathbb{R}^n} 2 \sum_{i=1}^n \alpha_i y_i - \sum_{i,j=1}^n \alpha_i \alpha_j K(x_i, x_j) = 2\alpha^\top y - \alpha^\top K\alpha,$$

subject to:

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

- The complimentary slackness conditions are, for \( i = 1, \ldots, n \):

\[
\begin{aligned}
\mu_i \left[ y_i f(x_i) + \xi_i - 1 \right] &= 0, \\
\nu_i \xi_i &= 0,
\end{aligned}
\]

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

\[
\begin{aligned}
\alpha_i \left[ y_i f(x_i) + \xi_i - 1 \right] &= 0, \\
\left( \alpha_i - \frac{y_i}{2\lambda n} \right) \xi_i &= 0.
\end{aligned}
\]
Analysis of KKT conditions

\[
\begin{cases}
\alpha_i [y_if(x_i) + \xi_i - 1] = 0, \\
(\alpha_i - \frac{y_i}{2\lambda n}) \xi_i = 0.
\end{cases}
\]

- If \( \alpha_i = 0 \), then the second constraint is active: \( \xi_i = 0 \). This implies \( y_if(x_i) \geq 1 \).
- If \( 0 < y_i\alpha_i < \frac{1}{2\lambda n} \), then both constraints are active: \( \xi_i = 0 \) et \( y_if(x_i) + \xi_i - 1 = 0 \). This implies \( y_if(x_i) = 1 \).
- 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: \( y_if(x_i) + \xi_i = 1 \). This implies \( y_if(x_i) \leq 1 \).
Another point of view without KKT

The dual can be rewritten as the minimization of a quadratic function under box constraints

\[
\min_{\alpha \in \mathbb{R}^n} \left\{ q(\alpha) = \frac{1}{2} \alpha^T K \alpha - \alpha^T y \right\} \quad \text{s.t.} \quad \forall i, \ 0 \leq y_i \alpha_i \leq C,
\]

The gradient is \( \nabla q(\alpha) = K\alpha - y = [f(x_i) - y_i]_{i=1,...,n} \).

Assume \( y_i = 1 \) (case with \( y_i = -1 \) is similar) and consider three cases:

- **Case 1:** \( 0 < y_i \alpha_i^* < C \);
- \( [\nabla q(\alpha^*)]_i = 0 \);
- \( \Rightarrow y_i f(x_i) = 1 \).
Another point of view without KKT

The dual can be rewritten as the minimization of a quadratic function under box constraints

$$\min_{\alpha \in \mathbb{R}^n} \left\{ q(\alpha) = \frac{1}{2} \alpha^\top K \alpha - \alpha^\top y \right\} \quad \text{s.t. \ } \forall i, \ 0 \leq y_i \alpha_i \leq C,$$

The gradient is $\nabla q(\alpha) = K\alpha - y = [f(x_i) - y_i]_{i=1,...,n}$.

Assume $y_i = 1$ (case with $y_i = -1$ is similar) and consider three cases:

- **Case 2**: $y_i \alpha_i^* = C$;
- $[\nabla q(\alpha^*)]_i \leq 0$;
- $\Rightarrow y_i f(x_i) \leq 1$. 

![Graph of $q(\alpha)$ with points $\alpha_i^*$ and $C$.](image-url)
Another point of view without KKT

The dual can be rewritten as the minimization of a quadratic function under box constraints

\[
\min_{\alpha \in \mathbb{R}^n} \left\{ q(\alpha) = \frac{1}{2} \alpha^T K \alpha - \alpha^T y \right\} \quad \text{s.t.} \quad \forall i, \ 0 \leq y_i \alpha_i \leq C,
\]

The gradient is \( \nabla q(\alpha) = K\alpha - y = [f(x_i) - y_i]_{i=1,\ldots,n} \).

Assume \( y_i = 1 \) (case with \( y_i = -1 \) is similar) and consider three cases:

- **Case 3:** \( \alpha_i^* = 0; \)
- \( [\nabla q(\alpha^*)]_i \geq 0; \)
- \( \Rightarrow y_i f(x_i) \geq 1. \)
Geometric interpretation
Geometric interpretation

- $f(x) = +1$
- $f(x) = 0$
- $f(x) = -1$
Geometric interpretation

$\alpha y = 1/2n\lambda$

$0 < \alpha y < 1/2n\lambda$

$\alpha = 0$
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:

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

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:

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

- This is equivalent to our formulation with $C = \frac{1}{2n\lambda}$.

- The SVM optimization problem is then:

$$\max_{\alpha \in \mathbb{R}^d} 2 \sum_{i=1}^{n} \alpha_i y_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j),$$

subject to:

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

- 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:

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

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

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

subject to:

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

This is therefore equivalent to the previous SVM with the kernel $K + n\lambda I$ and $C = +\infty$. 
Kernel Methods
Unsupervised Learning
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
   - Kernel PCA
     - Kernel K-means and spectral clustering
     - A quick note on kernel CCA
5. The Kernel Jungle
6. Open Problems and Research Topics
Principal Component Analysis (PCA)

Classical setting

- Let $S = \{x_1, \ldots, x_n\}$ be a set of vectors ($x_i \in \mathbb{R}^d$)
- 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
Formalization

- Assume that the data are centered (otherwise center them as preprocessing), i.e.:
  \[
  \frac{1}{n} \sum_{i=1}^{n} x_i = 0.
  \]

- The orthogonal projection onto a direction \( w \in \mathbb{R}^d \) is the function \( h_w : \mathbb{R}^d \to \mathbb{R} \) defined by:
  \[
  h_w (x) = x^\top \frac{w}{\|w\|}.
  \]
Principal Component Analysis (PCA)

Formalization

- The empirical variance captured by $h_w$ is:

$$\hat{\text{var}}(h_w) := \frac{1}{n} \sum_{i=1}^{n} h_w(x_i)^2 = \frac{1}{n} \sum_{i=1}^{n} \frac{(x_i^\top w)^2}{\|w\|^2}.$$

- The $i$-th principal direction $w_i$ ($i = 1, \ldots, d$) is defined by:

$$w_i = \arg \max_{w \perp \{w_1, \ldots, w_{i-1}\}} \hat{\text{var}}(h_w) \quad \text{s.t.} \quad \|w\| = 1.$$
Principal Component Analysis (PCA)

**Solution**

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

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

- The solutions of:

\[
\mathbf{w}_i = \arg \max_{\mathbf{w} \perp \{\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}\}} \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}w \quad \text{s.t.} \quad \|\mathbf{w}\| = 1
\]
Let $X$ be the $n \times d$ data matrix whose rows are the vectors $x_1, \ldots, x_n$. We can then write:

$$\hat{\text{var}}(h_w) = \frac{1}{n} \sum_{i=1}^{n} \frac{(x_i^\top w)^2}{\|w\|^2} = \frac{1}{n} \frac{w^\top X^\top X w}{w^\top w}.$$ 

The solutions of:

$$w_i = \arg \max_{w \perp \{w_1, \ldots, w_{i-1}\}} w^\top X^\top X w \text{ s.t. } \|w\| = 1$$

are the successive eigenvectors of $X^\top X$, ranked by decreasing eigenvalues.
Kernel Principal Component Analysis (PCA)

Let \( \mathbf{x}_1, \ldots, \mathbf{x}_n \) be a set of data points in \( \mathcal{X} \); let \( K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) be a positive definite kernel and \( \mathcal{H} \) be its RKHS.

**Formalization**

- Assume that the data are centered (otherwise center by manipulating the kernel matrix), i.e.:
  \[
  \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \implies \frac{1}{n} \sum_{i=1}^{n} \varphi(\mathbf{x}_i) = 0.
  \]

- The orthogonal projection onto a direction \( f \in \mathcal{H} \) is the function \( h_f : \mathcal{X} \rightarrow \mathbb{R} \) defined by:
  \[
  h_f(\mathbf{x}) = \frac{\mathbf{x}^\top \mathbf{w}}{\|\mathbf{w}\|} \implies h_f(\mathbf{x}) = \left\langle \varphi(\mathbf{x}), \frac{f}{\|f\|_\mathcal{H}} \right\rangle_\mathcal{H}.
  \]
Kernel Principal Component Analysis (PCA)

Let \( x_1, \ldots, x_n \) be a set of data points in \( \mathcal{X} \); let \( K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) be a positive definite kernel and \( \mathcal{H} \) be its RKHS.

**Formalization**

- The empirical variance captured by \( h_f \) is:

  \[
  \hat{\mathrm{var}} (h_w) = \frac{1}{n} \sum_{i=1}^{n} \frac{(x_i^\top w)^2}{\| w \|^2} \quad \Rightarrow \quad \hat{\mathrm{var}} (h_f) := \frac{1}{n} \sum_{i=1}^{n} \frac{\langle \varphi(x_i), f \rangle^2_{\mathcal{H}}}{\| f \|^2_{\mathcal{H}}}. 
  \]

- The \( i \)-th principal direction \( f_i \) (\( i = 1, \ldots, d \)) is defined by:

  \[
  f_i = \arg \max_{f \perp \{ f_1, \ldots, f_{i-1} \}} \hat{\mathrm{var}} (h_f) \quad \text{s.t.} \quad \| f \|_{\mathcal{H}} = 1.
  \]
Kernel Principal Component Analysis (PCA)

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a set of data points in $\mathcal{X}$; let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel and $\mathcal{H}$ be its RKHS.

Formalization

- The empirical variance captured by $h_f$ is:

$$\hat{\text{var}}(h_w) = \frac{1}{n} \sum_{i=1}^{n} \frac{(\mathbf{x}_i^\top \mathbf{w})^2}{\| \mathbf{w} \|^2} \implies \hat{\text{var}}(h_f) := \frac{1}{n} \sum_{i=1}^{n} \frac{f(\mathbf{x}_i)^2}{\| f \|^2_{\mathcal{H}}}. $$

- The $i$-th principal direction $f_i$ ($i = 1, \ldots, d$) is defined by:

$$f_i = \arg \max_{f \perp \{f_1, \ldots, f_{i-1}\}} \sum_{i=1}^{n} f(\mathbf{x}_i)^2 \text{ s.t. } \| f \|_{\mathcal{H}} = 1.$$
Let \( K(x, y) = x^\top y \) be the linear kernel.

The associated RKHS \( \mathcal{H} \) is the set of linear functions:

\[
f_w(x) = w^\top x,
\]

endowed with the norm \( \| f_w \|_{\mathcal{H}} = \| w \|_{\mathbb{R}^d} \).

Therefore we can write:

\[
\hat{\text{var}}(h_w) = \frac{1}{n} \sum_{i=1}^{n} \left( x_i^\top w \right)^2 = \frac{1}{n \| f_w \|^2} \sum_{i=1}^{n} f_w(x_i)^2.
\]

Moreover, \( w \perp w' \Leftrightarrow f_w \perp f_{w'} \).
Kernel Principal Component Analysis (PCA)

Solution

Kernel PCA solves, for $i = 1, \ldots, d$:

$$f_i = \arg \max_{f \perp \{f_1, \ldots, f_{i-1}\}} \sum_{i=1}^{n} f(x_i)^2 \quad \text{s.t.} \quad \|f\|_{H} = 1.$$ 

We can apply the representer theorem (exercise: check that it is also valid in this case): for $i = 1, \ldots, d$, we have:

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

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

Kernel Principal Component Analysis (PCA)

- Therefore we have:

\[ \| f_i \|_H^2 = \sum_{k,l=1}^{n} \alpha_i,k \alpha_i,l K(x_k, x_l) = \alpha_i^\top K \alpha_i, \]

- Similarly:

\[ \sum_{k=1}^{n} f_i(x_k)^2 = \alpha_i^\top K^2 \alpha_i. \]

- and

\[ \langle f_i, f_j \rangle_H = \alpha_i^\top K \alpha_j. \]
Kernel Principal Component Analysis (PCA)

Solution

Kernel PCA maximizes in $\alpha$ the function:

$$\alpha_i = \arg \max_{\alpha \in \mathbb{R}^n} \alpha^\top K^2 \alpha,$$

under the constraints:

$$\begin{cases} 
\alpha_i^\top K \alpha_j &= 0 \quad \text{for } j = 1, \ldots, i - 1. \\
\alpha_i^\top K \alpha_i &= 1
\end{cases}$$
Kernel Principal Component Analysis (PCA)

Solution

- Compute the eigenvalue decomposition of the kernel matrix $K = U\Delta U^\top$, with eigenvalues $\Delta_1 \geq \ldots \geq \Delta_n \geq 0$.
- After a change of variable $\beta = K^{1/2}\alpha$ (with $K^{1/2} = U\Delta^{1/2}U^\top$),
  \[
  \beta_i = \arg \max_{\beta \in \mathbb{R}^n} \beta^\top K \beta,
  \]
  under the constraints:
  \[
  \begin{align*}
  \beta_i^\top \beta_j &= 0 \quad \text{for } j = 1, \ldots, i - 1, \\
  \beta_i^\top \beta_i &= 1
  \end{align*}
  \]
- Thus, $\beta_i = u_i$ ($i$-th eigenvector) is a solution!
- Finally, $\alpha_i = \frac{1}{\sqrt{\Delta_i}} u_i$. 
Kernel Principal Component Analysis (PCA)

Summary

1. Center the Gram matrix
2. Compute the first eigenvectors \((u_i, \Delta_i)\)
3. Normalize the eigenvectors \(\alpha_i = u_i / \sqrt{\Delta_i}\)
4. The projections of the points onto the \(i\)-th eigenvector is given by \(K\alpha_i\)
Kernel Principal Component Analysis (PCA)

Remarks

- In this formulation, we must **diagonalize the centered kernel Gram matrix**, instead of the covariance matrix in the classical setting.

- **Exercise:** check that $\mathbf{X}^\top \mathbf{X}$ and $\mathbf{XX}^\top$ 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...
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. Kernel tricks
3. Kernel Methods: Supervised Learning
   - Kernel PCA
   - Kernel K-means and spectral clustering
   - A quick note on kernel CCA
5. The Kernel Jungle
6. Open Problems and Research Topics
The K-means algorithm

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

**Optimization point of view**

Given data points $x_1, \ldots, x_n$ in $\mathbb{R}^p$, it consists of performing alternate minimization steps for optimizing the following cost function

$$\min_{\mu_j \in \mathbb{R}^p} \sum_{i=1}^{n} \sum_{j=1}^{k} \|x_i - \mu_{s_i}\|^2_2.$$  

K-means alternates between two steps:

1. **cluster assignment:**

   Given fixed $\mu_1, \ldots, \mu_k$, assign each $x_i$ to its closest centroid

   $$\forall i, \ s_i \in \arg\min_{s \in \{1, \ldots, k\}} \|x_i - \mu_s\|^2_2.$$
The K-means algorithm

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

Optimization point of view

Given data points $x_1, \ldots, x_n$ in $\mathbb{R}^p$, it consists of performing alternate minimization steps for optimizing the following cost function

$$\min_{\mu_j \in \mathbb{R}^p} \sum_{i=1}^{n} \left\| x_i - \mu_{s_i} \right\|^2_2.$$  

K-means alternates between two steps:

2. **centroids update:**

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

   $$\forall j, \quad \mu_j = \arg\min_{\mu \in \mathbb{R}^p} \sum_{i: s_i = j} \left\| x_i - \mu \right\|^2_2.$$
The K-means algorithm

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

Optimization point of view

Given data points $x_1, \ldots, x_n$ in $\mathbb{R}^p$, it consists of performing alternate minimization steps for optimizing the following cost function

$$
\min_{\mu_j \in \mathbb{R}^p} \sum_{i=1}^{n} \|x_i - \mu_{s_i}\|_2^2.
$$

\(s_i \in \{1, \ldots, k\}, \text{ for } i=1, \ldots, n\)

K-means alternates between two steps:

2 centroids update:

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

$$
\iff \forall j, \quad \mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i \quad \text{with} \quad C_j = \{i : s_i = j\}.
$$
The kernel K-means algorithm

We may now modify the objective to operate in a RKHS. Given data points $x_1, \ldots, x_n$ in $\mathcal{X}$ and a p.d. kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with $\mathcal{H}$ its RKHS, the new objective becomes

$$\min_{\mu_j \in \mathcal{H} \text{ for } j = 1, \ldots, k} \sum_{i = 1}^{n} \| \phi(x_i) - \mu_s \|^2_H.$$
The kernel K-means algorithm

We may now modify the objective to operate in a RKHS. Given data points $x_1, \ldots, x_n$ 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

$$\min_{\mu_j \in \mathcal{H}} \sum_{i=1}^{n} \sum_{s_i \in \{1, \ldots, k\}} \|\varphi(x_i) - \mu_{s_i}\|^2_{\mathcal{H}}.$$

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

**Proposition**

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

$$\min_{\mu \in \mathcal{H}} \sum_{i=1}^{n} \|\varphi(x_i) - \mu\|^2_{\mathcal{H}}.$$
The kernel K-means algorithm

Proof

\[
\frac{1}{n} \sum_{i=1}^{n} \| \varphi(x_i) - \mu \|^2_H = \frac{1}{n} \sum_{i=1}^{n} \| \varphi(x_i) \|^2_H - \left\langle \frac{2}{n} \sum_{i=1}^{n} \varphi(x_i), \mu \right\rangle_H + \| \mu \|^2_H \\
= \frac{1}{n} \sum_{i=1}^{n} \| \varphi(x_i) \|^2_H - 2 \left\langle \varphi_n, \mu \right\rangle_H + \| \mu \|^2_H \\
= \frac{1}{n} \sum_{i=1}^{n} \| \varphi(x_i) \|^2_H - \| \varphi_n \|^2_H + \| \varphi_n - \mu \|^2_H,
\]

which is minimum for \( \mu = \varphi_n \).
The kernel K-means algorithm

Given now the objective,

\[
\min_{\mu_j \in \mathcal{H}} \sum_{i=1}^{n} \left\| \varphi(x_i) - \mu_{s_i} \right\|_H^2,
\]

we know that given assignments \( s_i \), the optimal \( \mu_j \) are the centers of mass of the respective clusters and we obtain

**Greedy approach: kernel K-means**

We alternate between two steps:

1. **centroids update:**
   Given the previous assignments \( s_1, \ldots, s_n \), update the centroids

\[
\forall j, \quad \mu_j = \arg\min_{\mu \in \mathcal{H}} \sum_{i : s_i = j} \left\| \varphi(x_i) - \mu \right\|_H^2.
\]
The kernel K-means algorithm

Given now the objective,

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

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

Greedy approach: kernel K-means

We alternate between two steps:

1 centroids update:
   Given the previous assignments $s_1, \ldots, s_n$, update the centroids

$$\Leftrightarrow \forall j, \quad \mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} \varphi(x_i).$$
The kernel K-means algorithm

Given now the objective,

$$\min_{\mu_j \in \mathcal{H}} \sum_{i=1}^{n} \sum_{j=1}^{k} \left\| \varphi(x_i) - \mu_{s_i} \right\|^2_{\mathcal{H}},$$

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

**Greedy approach: kernel K-means**

We alternate between two steps:

2. **cluster assignment:**
   Given fixed $\mu_1, \ldots, \mu_k$, assign each $x_i$ to its closest centroid: $\forall \ i,$

   $$s_i \in \arg\min_{s \in \{1, \ldots, k\}} \left\| \varphi(x_i) - \mu_s \right\|^2_{\mathcal{H}}.$$
The kernel K-means algorithm

Given now the objective,

\[
\min_{\mu_j \in \mathcal{H}} \quad \text{for } j = 1, \ldots, k
\sum_{i=1}^{n} \| \varphi(x_i) - \mu_{s_i} \|^2_{\mathcal{H}},
\]

we know that given assignments \(s_i\), the optimal \(\mu_j\) are the centers of mass of the respective clusters and we obtain

**Greedy approach: kernel K-means**

We alternate between two steps:

1. **cluster assignment:**
   Given fixed \(\mu_1, \ldots, \mu_k\), assign each \(x_i\) to its closest centroid: \(\forall i, \)

   \[
   s_i \in \arg\min_{s \in \{1, \ldots, k\}} \left\| \varphi(x_i) - \frac{1}{|C_s|} \sum_{j \in C_s} \varphi(x_j) \right\|^2_{\mathcal{H}}
   \]

   \((C_s \text{ is from step 1}).\)
The kernel K-means algorithm

Given now the objective,

\[ \min_{\mu_j \in \mathcal{H}} \sum_{i=1}^{n} \| \varphi(x_i) - \mu_{s_i} \|^2_{\mathcal{H}}, \]

we know that given assignments \( s_i \), the optimal \( \mu_j \) are the centers of mass of the respective clusters and we obtain

**Greedy approach: kernel K-means**

We alternate between two steps:

1. **cluster assignment:**
   Given fixed \( \mu_1, \ldots, \mu_k \), assign each \( x_i \) to its closest centroid: \( \forall \ i, \)

   \[ s_i \in \arg\min_{s \in \{1, \ldots, k\}} \left( K(x_i, x_i) - \frac{2}{|C_s|} \sum_{j \in C_s} K(x_i, x_j) + \frac{1}{|C_s|^2} \sum_{j, l \in C_s} K(x_j, x_l) \right). \]
The kernel K-means algorithm, equivalent objective

Note that all operations are performed by manipulating kernel values \( K(x_i, x_j) \) only. Implicitly, we are optimizing in fact

\[
\min_{s_i \in \{1, \ldots, k\}} \sum_{i=1}^{n} \left\| \varphi(x_i) - \frac{1}{|C_{s_i}|} \sum_{j \in C_{s_i}} \varphi(x_j) \right\|_2^2,
\]

or, equivalently,

\[
\min_{s_i \in \{1, \ldots, k\}} \sum_{i=1}^{n} \left( K(x_i, x_i) - \frac{2}{|C_{s_i}|} \sum_{j \in C_{s_i}} K(x_i, x_j) + \frac{1}{|C_{s_i}|^2} \sum_{j, l \in C_{s_i}} K(x_j, x_l) \right).
\]

Then, notice that

\[
\sum_{i=1}^{n} \frac{1}{|C_{s_i}|^2} \sum_{j, l \in C_{s_i}} K(x_j, x_l) = \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{i, j \in C_l} K(x_i, x_j).
\]
The kernel $K$-means algorithm, equivalent objective

Note that all operations are performed by manipulating kernel values $K(x_i, x_j)$ only. Implicitly, we are optimizing in fact

$$\min_{s_i \in \{1, \ldots, k\}} \sum_{i=1}^{n} \left\| \varphi(x_i) - \frac{1}{|C_{s_i}|} \sum_{j \in C_{s_i}} \varphi(x_j) \right\|_2^2,$$

or, equivalently,

$$\min_{s_i \in \{1, \ldots, k\}} \sum_{i=1}^{n} \left( K(x_i, x_i) - \frac{2}{|C_{s_i}|} \sum_{j \in C_{s_i}} K(x_i, x_j) + \frac{1}{|C_{s_i}|^2} \sum_{j, l \in C_{s_i}} K(x_j, x_l) \right).$$

and

$$\sum_{i=1}^{n} \frac{1}{|C_{s_i}|} \sum_{j \in C_{s_i}} K(x_i, x_j) = \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{i, j \in C_l} K(x_i, x_j).$$
The kernel K-means algorithm, equivalent objective

Then, after removing the constant terms $K(x_i, x_i)$, we obtain:

**Proposition**

The kernel K-means objective is equivalent to the following one:

$$\max_{s_i \in \{1, \ldots, k\}} \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{i, j \in C_l} K(x_i, x_j).$$

This is a hard combinatorial optimization problem.

There are two types of algorithms to address it:
The kernel K-means algorithm, equivalent objective

Then, after removing the constant terms $K(x_i, x_i)$, we obtain:

**Proposition**

The kernel K-means objective is equivalent to the following one:

$$
\max_{s_i \in \{1, \ldots, k\}} \sum_{i=1}^{k} \frac{1}{|C_l|} \sum_{i,j \in C_l} K(x_i, x_j).
$$

This is a hard combinatorial optimization problem.

There are two types of algorithms to address it:

1. **greedy algorithm:** kernel K-means
2. **spectral relaxation:** spectral clustering
Spectral clustering algorithms

Instead of a greedy approach, we can relax the problem into a feasible one, which yields a class of algorithms called spectral clustering.

First, consider the objective

$$\max_{s_i \in \{1, \ldots, k\}} \frac{1}{|C_l|} \sum_{i,j \in C_l} K(x_i, x_j).$$

and we introduce

(*) the binary assignment matrix $A$ in $\{0, 1\}^{n \times k}$ whose rows sum to one.

(**) the diagonal rescaling matrix $D$ in $\mathbb{R}^{k \times k}$ with diagonal entries $[D]_{jj}$ equal to $(\sum_{i=1}^{n} [A]_{ij})^{-1}$: the inverse of the cardinality of cluster $j$.

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

$$\max_{A,D} \left[ \text{trace} \left( D^{1/2} A^\top KAD^{1/2} \right) \right] \quad \text{s.t. \ (\star) and \ (\star\star).}$$
Spectral clustering algorithms

\[ \max_{A,D} \text{trace}(D^{1/2}A^\top KAD^{1/2}) \text{ s.t. (⋆) and (⋆⋆)}. \]

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

\[ \max_{Z \in \mathbb{R}^{n \times k}} \text{trace}(Z^\top KZ) \text{ s.t. } Z^\top Z = I. \]
Spectral clustering algorithms

\[
\begin{align*}
\max_{A,D} \text{trace} (D^{1/2} A^\top K A D^{1/2}) \quad \text{s.t. (⋆) and (⋆⋆).}
\end{align*}
\]

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

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

A solution \( Z^* \) to this problem may be obtained by computing the eigenvectors of \( K \) associated to the \( k \)-largest eigenvalues. This procedure is related to the kernel PCA algorithm!

**Question**

How do we obtain an approximate solution \((A, D)\) of the original problem from the exact solution of the relaxed one \( Z^* \)?
Spectral clustering algorithms

\[
\max_{A,D} \text{trace} \left( \mathbf{D}^{1/2} \mathbf{A}^\top \mathbf{K} \mathbf{A} \mathbf{D}^{1/2} \right) \quad \text{s.t. } (\star) \text{ and } (\star\star).
\]

The constraints on \( A, D \) 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 (\( \star, \star\star \)) on \( A \) and \( D \) and instead optimize over \( \mathbf{Z} = \mathbf{A} \mathbf{D}^{1/2} \):

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

A solution \( \mathbf{Z}^\star \) to this problem may be obtained by computing the eigenvectors of \( \mathbf{K} \) associated to the \( k \)-largest eigenvalues. This procedure is related to the kernel PCA algorithm!

**Answer 1**

With the original constraints on \( A \), every row of \( A \) has a single non-zero entry \( \implies \) compute the maximum entry of every row of \( \mathbf{Z}^\star \).
Spectral clustering algorithms

\[
\max_{A,D} \text{trace} \left( D^{1/2} A^\top K A D^{1/2} \right) \quad \text{s.t. (⋆) and (⋆⋆).}
\]

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

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

A solution \( Z^* \) to this problem may be obtained by computing the eigenvectors of \( K \) associated to the \( k \)-largest eigenvalues. This procedure is related to the kernel PCA algorithm!

**Answer 2**

Normalize the rows of \( Z^* \) to have unit \( \ell_2 \)-norm, and apply the traditional \( K \)-means algorithm on the rows. This is called spectral clustering.
The constraints on $A, D$ are such that $D^{1/2}A^\top AD^{1/2} = I$ (exercise). A natural relaxation consists of dropping the constraints $(\star, \star\star)$ on $A$ and $D$ and instead optimize over $Z = AD^{1/2}$:

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

A solution $Z^*$ to this problem may be obtained by computing the eigenvectors of $K$ associated to the $k$-largest eigenvalues. This procedure is related to the kernel PCA algorithm!

**Answer 3**

Choose another variant of the previous procedures.
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
   - Kernel PCA
   - Kernel K-means and spectral clustering
   - A quick note on kernel CCA
5. The Kernel Jungle
6. Open Problems and Research Topics
Canonical Correlation Analysis (CCA)

Given two views $X = [x_1, \ldots, x_n]$ in $\mathbb{R}^{p \times n}$ and $Y = [y_1, \ldots, y_n]$ 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

$$\max_{w_a \in \mathbb{R}^p, w_b \in \mathbb{R}^d} \frac{\frac{1}{n} \sum_{i=1}^{n} w_a^\top x_i y_i w_b}{\left(\frac{1}{n} \sum_{i=1}^{n} w_a^\top x_i x_i^\top w_a\right)^{1/2} \left(\frac{1}{n} \sum_{i=1}^{n} w_b^\top y_i y_i^\top w_b\right)^{1/2}}.$$

Assuming that the pairs $(x_i, y_i)$ are i.i.d. samples from an unknown distribution, CCA seeks to maximize

$$\max_{w_a \in \mathbb{R}^p, w_b \in \mathbb{R}^d} \frac{\text{cov}(w_a^\top X, w_b^\top Y)}{\sqrt{\text{var}(w_a^\top X)} \sqrt{\text{var}(w_b^\top Y)}}.$$
Canonical Correlation Analysis (CCA)

Given two views $X = [x_1, \ldots, x_n]$ in $\mathbb{R}^{p \times n}$ and $Y = [y_1, \ldots, y_n]$ 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

$$
\max_{w_a \in \mathbb{R}^p, w_b \in \mathbb{R}^d} \frac{\frac{1}{n} \sum_{i=1}^{n} w_a^\top x_i y_i^\top w_b}{\left(\frac{1}{n} \sum_{i=1}^{n} w_a^\top x_i x_i^\top w_a\right)^{1/2} \left(\frac{1}{n} \sum_{i=1}^{n} w_b^\top y_i y_i^\top w_b\right)^{1/2}}.
$$

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

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.
Canonical Correlation Analysis (CCA)

Formulation

Assuming that the datasets are centered,

\[
\max_{\mathbf{w}_a \in \mathbb{R}^p, \mathbf{w}_b \in \mathbb{R}^d} \frac{\mathbf{w}_a^\top \mathbf{X}^\top \mathbf{Y} \mathbf{w}_b}{(\mathbf{w}_a^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}_a)^{1/2} (\mathbf{w}_b^\top \mathbf{Y}^\top \mathbf{Y} \mathbf{w}_b)^{1/2}}.
\]

can be formulated, after removing the scaling ambiguity, as

\[
\max_{\mathbf{w}_a \in \mathbb{R}^p, \mathbf{w}_b \in \mathbb{R}^d} \mathbf{w}_a^\top \mathbf{X}^\top \mathbf{Y} \mathbf{w}_b \quad \text{s.t.} \quad \mathbf{w}_a^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}_a = 1 \quad \text{and} \quad \mathbf{w}_b^\top \mathbf{Y}^\top \mathbf{Y} \mathbf{w}_b = 1.
\]

Then, there exists \(\lambda_a\) and \(\lambda_b\) such that the problem is equivalent to

\[
\min_{\mathbf{w}_a \in \mathbb{R}^p, \mathbf{w}_b \in \mathbb{R}^d} -\mathbf{w}_a^\top \mathbf{X}^\top \mathbf{Y} \mathbf{w}_b + \frac{\lambda_a}{2} (\mathbf{w}_a^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}_a - 1) + \frac{\lambda_b}{2} (\mathbf{w}_b^\top \mathbf{Y}^\top \mathbf{Y} \mathbf{w}_b - 1).
\]
Canonical Correlation Analysis (CCA)

Taking the derivatives and setting the gradient to zero, we obtain

\[-X^\top Yw_b + \lambda_a X^\top Xw_a = 0\]
\[-Y^\top Xw_a + \lambda_b Y^\top Yw_b = 0\]

Multiply first equality by $w_a^\top$ and second equality by $w_b^\top$; subtract the two resulting equalities and we get

\[\lambda_a w_a^\top X^\top Xw_a = \lambda_b w_b^\top Y^\top Yw_b = \lambda_a = \lambda_b = \lambda,\]

and then, we obtain the generalized eigenvalue problem:

\[
\begin{bmatrix}
0 & X^\top Y \\
Y^\top X & 0
\end{bmatrix}
\begin{bmatrix}
w_a \\
w_b
\end{bmatrix}
= \lambda
\begin{bmatrix}
X^\top X & 0 \\
0 & Y^\top Y
\end{bmatrix}
\begin{bmatrix}
w_a \\
w_b
\end{bmatrix}
\]
Canonical Correlation Analysis (CCA)

Let us define

\[
\Sigma_A = \begin{bmatrix}
0 & X^\top Y \\
Y^\top X & 0
\end{bmatrix}, \quad \Sigma_B = \begin{bmatrix}
X^\top X & 0 \\
0 & Y^\top Y
\end{bmatrix} \quad \text{and} \quad w = \begin{bmatrix}
w_a \\
w_b
\end{bmatrix}
\]

Assuming the covariances are invertible, the generalized eigenvalue problem is equivalent to

\[
\Sigma_B^{-1/2} \Sigma_A w = \lambda \Sigma_B^{1/2} w
\]

which is also equivalent to the eigenvalue problem

\[
\Sigma_B^{-1/2} \Sigma_A \Sigma_B^{-1/2} (\Sigma_B^{1/2} w) = \lambda (\Sigma_B^{1/2} w).
\]
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 $x_1, \ldots, x_n$ in $\mathcal{X}^n$:

$$(\varphi_a(x_1), \ldots, \varphi_a(x_n)) \quad \text{and} \quad (\varphi_b(x_1), \ldots, \varphi_b(x_n)),$$

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.

**Formulation**

Then, we may formulate kernel CCA as

$$\max_{f_a \in \mathcal{H}_a, f_b \in \mathcal{H}_b} \frac{\frac{1}{n} \sum_{i=1}^{n} \langle f_a, \varphi_a(x_i) \rangle_{\mathcal{H}_a} \langle \varphi_b(x_i), f_b \rangle_{\mathcal{H}_b}}{\left( \frac{1}{n} \sum_{i=1}^{n} \langle f_a, \varphi_a(x_i) \rangle_{\mathcal{H}_a}^2 \right)^{1/2} \left( \frac{1}{n} \sum_{i=1}^{n} \langle f_b, \varphi_b(x_i) \rangle_{\mathcal{H}_b}^2 \right)^{1/2}}.$$
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 \( x_1, \ldots, x_n \) in \( \mathcal{X}^n \):

\[
(\varphi_a(x_1), \ldots, \varphi_a(x_n)) \quad \text{and} \quad (\varphi_b(x_1), \ldots, \varphi_b(x_n)),
\]

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.

**Formulation**

Then, we may formulate kernel CCA as

\[
\max_{f_a \in \mathcal{H}_a, f_b \in \mathcal{H}_b} \frac{\frac{1}{n} \sum_{i=1}^{n} f_a(x_i)f_b(x_i)}{\left( \frac{1}{n} \sum_{i=1}^{n} f_a(x_i)^2 \right)^{1/2} \left( \frac{1}{n} \sum_{i=1}^{n} f_b(x_i)^2 \right)^{1/2}}.
\]
Kernel Canonical Correlation Analysis

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

$$\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \frac{\frac{1}{n} \sum_{i=1}^{n} [K_a \alpha]_i [K_b \beta]_i}{(\frac{1}{n} \sum_{i=1}^{n} [K_a \alpha]_i^2)^{1/2}(\frac{1}{n} \sum_{i=1}^{n} [K_b \beta]_i^2)^{1/2}},$$

which is equivalent to

$$\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \frac{\alpha^\top K_a K_b \beta}{(\alpha^\top K_a^2 \alpha)^{1/2}(\beta^\top K_b^2 \beta)^{1/2}},$$

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

**Equivalent formulation**

$$\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \alpha^\top K_a K_b \beta \quad \text{s.t.} \quad \alpha^\top K_a^2 \alpha = 1 \quad \text{and} \quad \beta^\top K_b^2 \beta = 1.$$
Kernel Canonical Correlation Analysis

\[ \max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \alpha^T K_a K_b \beta \text{ s.t. } \alpha^T K_a^2 \alpha = 1 \quad \text{and} \quad \beta^T K_b^2 \beta = 1. \]

- This also leads to a generalized eigenvalue problem.
- The subsequent canonical directions are obtained by solving the same problem with additional orthogonality constraints.
Kernel Canonical Correlation Analysis

\[
\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \alpha^\top K_a K_b \beta \quad \text{s.t.} \quad \alpha^\top K_a^2 \alpha = 1 \quad \text{and} \quad \beta^\top K_b^2 \beta = 1.
\]

- This also leads to a generalized eigenvalue problem.
- The subsequent canonical directions are obtained by solving the same problem with additional orthogonality constraints.

What is wrong here?
Kernel Canonical Correlation Analysis

\[ \max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \alpha^\top K_a K_b \beta \quad \text{s.t.} \quad \alpha^\top K_a^2 \alpha = 1 \quad \text{and} \quad \beta^\top K_b^2 \beta = 1. \]

- This also leads to a generalized eigenvalue problem.
- The subsequent canonical directions are obtained by solving the same problem with additional orthogonality constraints.

What is wrong here?

If \( K_a \) and \( K_b \) are invertible, make the change of variable \( \alpha' = K_a \alpha \) and \( \beta' = K_b \beta \), and we obtain the equivalent formulation

\[ \max_{\alpha' \in \mathbb{R}^n, \beta' \in \mathbb{R}^n} \alpha'^\top \beta' \quad \text{s.t.} \quad \alpha'^\top \alpha' = 1 \quad \text{and} \quad \beta'^\top \beta' = 1. \]

The function is maximized for any \( \alpha' = \beta' \) in \( \mathbb{R}^n \).
Kernel Canonical Correlation Analysis

\[
\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \alpha^\top K_a K_b \beta \quad \text{s.t.} \quad \alpha^\top K_a^2 \alpha = 1 \quad \text{and} \quad \beta^\top K_b^2 \beta = 1.
\]

- This also leads to a generalized eigenvalue problem.
- The subsequent canonical directions are obtained by solving the same problem with additional orthogonality constraints.

What is wrong here?

If \( K_a \) and \( K_b \) are invertible, make the change of variable \( \alpha' = K_a \alpha \) and \( \beta' = K_b \beta \), and we obtain the equivalent formulation

\[
\max_{\alpha' \in \mathbb{R}^n, \beta' \in \mathbb{R}^n} \alpha'^\top \beta' \quad \text{s.t.} \quad \alpha'^\top \alpha' = 1 \quad \text{and} \quad \beta'^\top \beta' = 1.
\]

The function is maximized for any \( \alpha' = \beta' \) in \( \mathbb{R}^n \). In high (or infinite) dimension, it is easy to find spurious correlations.
Spurious correlations

Spurious correlations are bad:

Figure: http://www.tylervigen.com/.
Spurious correlations

Spurious correlations are bad:

Figure: [Link](http://www.tylervigen.com/).
Kernel Canonical Correlation Analysis

\[
\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \alpha^T K_a K_b \beta \quad \text{s.t.} \quad \alpha^T K_a^2 \alpha = 1 \quad \text{and} \quad \beta^T K_b^2 \beta = 1.
\]

- spurious correlation is a problem of overfitting;
- it also a problem of numerical instability, due to the need to invert the kernel matrices;
Kernel Canonical Correlation Analysis

\[
\max_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^n} \alpha^\top K_a K_b \beta \quad \text{s.t.} \quad \alpha^\top K_a^2 \alpha = 1 \quad \text{and} \quad \beta^\top K_b^2 \beta = 1.
\]

- spurious correlation is a problem of overfitting;
- it also a problem of numerical instability, due to the need to invert the kernel matrices;

A solution to both problems: Regularize!

- Find smooth directions \((f_a, f_b)\) by penalizing \(\|f_a\|_{\mathcal{H}_a}\) and \(\|f_b\|_{\mathcal{H}_b}\).
- it consists of replacing the constraints \(\alpha^\top K_a^2 \alpha = 1\) by

\[
(1 - \tau) \alpha^\top K_a^2 \alpha + \tau \underbrace{\alpha^\top K_a \alpha}_{\|f_a\|_{\mathcal{H}_a}^2} = 1,
\]

and do the same for \(\beta^\top K_b^2 \beta = 1\).
Application of kernel CCA

Finding a joint latent representation of text (tags) and images.

Figure: Figure from Gong and Lazebnik, 2014.
Part 5

The Kernel Jungle
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
5. The Kernel Jungle
   - Green, Mercer, Herglotz, Bochner and friends
     - Kernels for probabilistic models
     - Kernels for biological sequences
     - Kernels for graphs
     - Kernels on graphs
6. Open Problems and Research Topics
Introduction

- The kernel function plays a critical role in the **performance** of kernel methods.
- It is the place where **prior knowledge** about the problem can be inserted, in particular by controlling the norm of functions in the RKHS.
- In this part we provide some intuition about the link between kernels and smoothness functional through several examples.
- Subsequent parts will focus on the **design** of kernels for particular types of data.
The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
  - Green kernels
  - Mercer kernels
  - Convergence rates of KRR for Mercer kernels
  - Shift-invariant kernels
  - Generalization to semigroups
- Kernels for probabilistic 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).
- Example: spline regression

$$\min_f \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \int (f''(t))^2 \, dt$$

- In this section we make a general link between RKHS and Green functions defined by differential operators.
A simple example

Let

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

endowed with the bilinear form:

\[ \forall f, g \in \mathcal{H}, \quad \langle f, g \rangle_{\mathcal{H}} = \int_{0}^{1} f'(u) g'(u) \, du . \]

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 an RKHS with r.k. given by:

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

Therefore, the RKHS norm is precisely the smoothness functional defined in the simple example:

\[ \| f \|_{\mathcal{H}} = \| f' \|_{L^2([0,1])} \]

In particular, the following problem

\[ \min_{f \in \mathcal{H}} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_0^1 (f'(t))^2 \, dt \]

can be reformulated as a simple kernel ridge regression problem with kernel \( K(x, y) = \min(x, y) \):

\[ \min_{f \in \mathcal{H}} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \| f \|_{\mathcal{H}}^2 \]
Let $I \subset \mathbb{R}$ an interval. A function $f : I \rightarrow \mathbb{R}$ is absolutely continuous (AC) on $I$ if for any $\epsilon > 0$, there exists $\delta > 0$ such that for any finite sequence of pairwise disjoint sub-intervals $(u_k, v_k) \subset I$ such that \[ \sum_k (v_k - u_k) < \delta, \] it holds that \[ \sum_k |f(v_k) - f(u_k)| < \epsilon. \]

- AC $\implies$ uniformly continuous $\implies$ continuous
- $f$ AC on $[a, b]$ $\iff$ $f$ has derivative $f'$ almost everywhere, $f'$ is Lebesgue integrable, and for all $x \in [a, b]$ 
  \[ f(x) = f(a) + \int_a^x f'(t)dt \]
  $\iff$ there exists a Lebesgue integrable function $g$ on $[a, b]$ such that for all $x \in [a, b]$,
  \[ f(x) = f(a) + \int_a^x g(t)dt \]
  in which case $g = f'$ almost everywhere.
Proof (1/5)

We need to show that

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

Proof (2/5)

\( \mathcal{H} \) is a pre-Hilbert space of functions

- \( \mathcal{H} \) is a vector space of functions, and \( \langle f, g \rangle_{\mathcal{H}} \) a bilinear form that satisfies \( \langle f, f \rangle_{\mathcal{H}} \geq 0 \).
- \( f \) 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:

\[
|f(x)| = \left| \int_0^x f'(u)du \right| \leq \sqrt{x} \left( \int_0^1 (f'(u))^2du \right)^{1/2} = \sqrt{x} \langle f, f \rangle_{\mathcal{H}}^{1/2}.
\]

Therefore, \( \langle f, f \rangle_{\mathcal{H}} = 0 \implies f = 0, \) showing that \( \langle ., . \rangle_{\mathcal{H}} \) is an inner product. \( \mathcal{H} \) is thus a pre-Hilbert space.
**Proof (3/5)**

\( \mathcal{H} \) 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:

\[
\int_0^x f'_n(u)du = \int_0^x g(u)du,
\]

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

\[
\lim_n \| f_n - f \|_{\mathcal{H}} = \| f' - g_n \|_{L^2([0,1])} = 0.
\]
\( \forall x \in [0, 1], \ K_x \in \mathcal{H} \)

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

\[ \text{K}(s,t) \]

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

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

- This shows that $\mathcal{H}$ is a RKHS with $K$ as r.k. $\square$
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 $D$. 
Green function?

Definition

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 $x \in \mathcal{X}$,

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

If such a $k$ exists, it is called the **Green function** of the operator $D$. 
Proof

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

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

and $K$ be the Green function of the operator $D^*D$.

- For all $x \in \mathcal{X}$, $K_x \in \mathcal{H}$ because:

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

(caveat: sometimes other conditions must be fulfilled to be in $\mathcal{H}$, to be checked on a case by case basis).

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

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

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

- Back to our example, take $\mathcal{X} = [0, 1]$ and $Df(u) = f'(u)$
- To find the r.k. of $\mathcal{H}$ we need to solve in $k$: 
  \[
  f(x) = \langle D^* Dk_x, f \rangle_{L^2([0,1])} \\
  = \langle Dk_x, Df \rangle_{L^2([0,1])} \\
  = \int_0^1 k_x'(u)f'(u)\,du 
  \]
- The solution is 
  \[
  k_x'(u) = 1_{[0,x]}(u) 
  \]
  which gives 
  \[
  k_x(u) = \begin{cases} 
    u & \text{if } u \leq x, \\
    x & \text{otherwise.} 
  \end{cases} 
  \]
  and therefore 
  \[
  k(x, x') = \min(x, x') 
  \]
Outline

The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
  - Green kernels
  - Mercer kernels
  - Convergence rates of KRR for Mercer kernels
  - Shift-invariant kernels
  - Generalization to semigroups
- Kernels for probabilistic 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 that a Mercer kernel is an inner product

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:

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

where $\lambda_i \geq 0$ 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 (equivalence classes of) square integrable functions on $\mathcal{X}$.
- For any function $K : \mathcal{X}^2 \mapsto \mathbb{R}$, let the transform:

$$\forall f \in L^2_\nu(\mathcal{X}), \quad (L_K f)(x) = \int K(x, t) f(t) d\nu(t).$$

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 positive.
$L_K$ is a mapping from $L^2_\nu(\mathcal{X})$ to $L^2_\nu(\mathcal{X}')$.

For any $f \in L^2_\nu(\mathcal{X})$ and $(x_1, x_1) \in \mathcal{X}^2$:

$$
| (L_K f)(x_1) - (L_K f)(x_2) | = \left| \int (K(x_1, t) - K(x_2, t)) f(t) \, d\nu(t) \right|
$$

$$
= \langle Kx_1 - Kx_2, f \rangle_{L^2_\nu(\mathcal{X})}
$$

$$
\leq \| Kx_1 - Kx_2 \|_{L^2_\nu(\mathcal{X})} \| f \|_{L^2_\nu(\mathcal{X})}
$$

(Cauchy-Schwarz)

$$
\leq \sqrt{\nu(\mathcal{X})} \max_{t \in \mathcal{X}} | K(x_1, t) - K(x_2, t) | \| f \|_{L^2_\nu(\mathcal{X})}.
$$

$K$ being continuous and $\mathcal{X}$ compact, $K$ is uniformly continuous, therefore $L_K f$ is continuous. In particular, $L_K f \in L^2_\nu(\mathcal{X}')$ (with the slight abuse of notation $C(\mathcal{X'}) \subset L^2_\nu(\mathcal{X}')$). □
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 $x \in \mathcal{X}$:

\[
| (L_K f)(x) | = \left| \int K(x, t) f(t) \, d\nu(t) \right|
\]

\[
\leq \sqrt{\nu(\mathcal{X})} \max_{t \in \mathcal{X}} |K(x, t)| \| f \|_{L^2_\nu(\mathcal{X})}
\]

\[
\leq \sqrt{\nu(\mathcal{X})} C_K \| f \|_{L^2_\nu(\mathcal{X})}.
\]

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

\[
\| L_K f \|_{L^2_\nu(\mathcal{X})} = \left( \int (L_K f)(t)^2 \, d\nu(t) \right)^{\frac{1}{2}} \leq \nu(\mathcal{X}) C_K \| f \|_{L^2_\nu(\mathcal{X})}. \quad \square
\]
Proof (3/6)

**Criterion for compactness**

In order to prove the compactness of $L_K$ we need the following criterion. Let $C(X)$ denote the set of continuous functions on $X$ endowed with infinite norm $\| f \|_\infty = \max_{x \in X} |f(x)|$. A set of functions $G \subset C(X)$ is called **equicontinuous** if:

$$\forall \epsilon > 0, \exists \delta > 0, \forall (x, y) \in X^2,$$

$$\| x - y \| < \delta \implies \forall g \in G, |g(x) - g(y)| < \epsilon.$$ 

**Ascoli Theorem**

A part $H \subset C(X)$ 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 $L^2_\nu(X)$ ($\|f_n\|_{L^2_\nu(X)} < M$ for all $n$). The sequence $(L_K f_n)_{n \geq 0}$ is a sequence of continuous functions, uniformly bounded because:

$$\|L_K f_n\|_\infty \leq \sqrt{\nu(X)} C_K \|f_n\|_{L^2_\nu(X)} \leq \sqrt{\nu(X)} C_K M.$$  

It is equicontinuous because:

$$|L_K f_n(x_1) - L_K f_n(x_2)| \leq \sqrt{\nu(X)} \max_{t \in \mathcal{X}} |K(x_1, t) - K(x_2, t)| M.$$  

By Ascoli theorem, we can extract a sequence uniformly convergent in $C(X)$, and therefore in $L^2_\nu(X)$. □
**L_K is self-adjoint**

*K* being symmetric, we have for all \( f, g \in L^2_\nu(\mathcal{X}) \):

\[
\langle f, Lg \rangle_{L^2_\nu(\mathcal{X})} = \int f(x)(Lg)(x) \, d\nu(x)
\]

\[
= \int \int f(x)g(t)K(x,t) \, d\nu(x) \, d\nu(t) \quad \text{(Fubini)}
\]

\[
= \langle Lf, g \rangle_{L^2_\nu(\mathcal{X})}.
\]
Proof (6/6)

$L_K$ is positive

We can approximate the integral by finite sums:

$$\langle f, Lf \rangle_{L^2_\nu(\mathcal{X})} = \int \int f(x) f(t) K(x, t) \nu(dx) \nu(dt)$$

$$= \lim_{k \to \infty} \frac{\nu(\mathcal{X})}{k^2} \sum_{i,j=1}^{k} K(x_i, x_j) f(x_i) f(x_j)$$

$$\geq 0,$$

because $K$ is positive definite. □
Diagonalization of the operator

We need the following general result (e.g., Debnath and Mikusiński, 2005, Section 4.10)

**Spectral theorem**

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

**Remark**

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

$$\psi_k = \frac{1}{\lambda_k} L_K \psi_k.$$
Main result

Mercer’s Theorem

Let $\mathcal{X}$ be a compact metric space, $\nu$ a nondegenerate\(^{a}\) Borel measure on $\mathcal{X}$, and $K$ a continuous p.d. kernel. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ denote the nonnegative eigenvalues of $L_K$ and $(\psi_1, \psi_2, \ldots)$ the corresponding eigenfunctions. Then all functions $\psi_k$ are continuous, and for any $x, t \in \mathcal{X}$:

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

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

\(^{a}\)i.e., $\nu(U) > 0$ for any nonempty open set $U \subset \mathcal{X}$
For any $k \geq 1$ such that $\lambda_k > 0$, $\psi_k \in \mathcal{H}$ (RKHS of $K$) 

If $\lambda_k > 0$, we have 

$$
\forall x \in X, \quad \psi_k(x) = \frac{1}{\lambda_k} L_K \psi_k(x) 
$$

$$
= \frac{1}{\lambda_k} \int K(x, t) \psi_k(t) d\nu(t) 
$$

$$
= \lim_{n \to +\infty} \nu(X) \sum_{i=1}^{n} \frac{1}{\lambda_k n} K(x, t_i) \psi_k(t_i) 
$$

for a set $t_1, t_2, \ldots$ conveniently chosen. Besides, $h_n \in \mathcal{H}$ for any $n \in \mathbb{N}$ and, for any $n, m \in \mathbb{N}$,

$$
\langle h_n, h_m \rangle_{\mathcal{H}} = \nu(X)^2 \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{1}{\lambda_k^2 nm} \psi_k(t_i) \psi_k(t_j) K(t_i, t_j). 
$$
Proof of Mercer’s Theorem (2/6)

For any $k \geq 1$ such that $\lambda_k > 0$, $\psi_k \in \mathcal{H}$ (cont.)

Therefore,

$$\lim_{n,m \to +\infty} \langle h_n, h_m \rangle_{\mathcal{H}} = \frac{1}{\lambda_k^2} \int \int K(t, t') \psi_k(t) \psi_k(t') d\nu(t) d\nu(t') := R,$$

and

$$\| h_n - h_m \|_{\mathcal{H}}^2 = \langle h_n, h_n \rangle_{\mathcal{H}} + \langle h_m, h_m \rangle_{\mathcal{H}} - 2 \langle h_n, h_m \rangle_{\mathcal{H}} \xrightarrow{n,m \to \infty} R + R - 2R = 0.$$

$(h_n)_{n \in \mathbb{N}}$ is therefore a Cauchy sequence in $\mathcal{H}$, which converges to a function $h \in \mathcal{H}$. In particular, for any $x \in \mathcal{X}$,

$$h(x) = \lim_{n \to +\infty} h_n(x) = \psi_k(x),$$

and finally $\psi_k = h \implies \psi_k \in \mathcal{H}$. \qed
Proof of Mercer’s Theorem (3/6)

Let $i, j \geq 1$ such that $\lambda_i, \lambda_j > 0$. Then $\sqrt{\lambda_i} \psi_i, \sqrt{\lambda_j} \psi_j \in \mathcal{H}$ and

$$\left\langle \sqrt{\lambda_i} \psi_i, \sqrt{\lambda_j} \psi_j \right\rangle_{\mathcal{H}} = \left\langle \frac{1}{\sqrt{\lambda_i}} \int K_t \psi_i(t) d\nu(t), \psi_i, \sqrt{\lambda_j} \psi_j \right\rangle_{\mathcal{H}}$$

$$= \sqrt{\frac{\lambda_j}{\lambda_i}} \int \left\langle K_t, \psi_j \right\rangle_{\mathcal{H}} \psi_i(t) d\nu(t)$$

$$= \sqrt{\frac{\lambda_j}{\lambda_i}} \int \psi_j(t) \psi_i(t) d\nu(t)$$

$$= \sqrt{\frac{\lambda_j}{\lambda_i}} \left\langle \psi_i, \psi_j \right\rangle_{L^2_\nu(X)}$$

$$= \delta_{i,j}.$$
Proof of Mercer’s Theorem (4/6)

For any \( x \in \mathcal{X} \), \( \sum_{k : \lambda_k > 0} \lambda_k \psi_k(x)^2 \leq C_K \)

For any \( x \in \mathcal{X} \), \( K_x \in \mathcal{H} \) and \( \|K_x\|_{\mathcal{H}}^2 = K(x, x) \leq C_K \).

Therefore, since \( \{ \sqrt{\lambda_k} \psi_k : \lambda_k > 0 \} \) is an ONS of \( \mathcal{H} \):

\[
C_K \geq \|K_x\|_{\mathcal{H}}^2 \\
\geq \sum_{k : \lambda_k > 0} \left\langle K_x, \sqrt{\lambda_k} \psi_k \right\rangle_{\mathcal{H}}^2 \\
= \sum_{k : \lambda_k > 0} \lambda_k \psi_k(x)^2. \quad \square
\]
Proof of Mercer’s Theorem (5/6)

For any $x \in \mathcal{X}$, $t \rightarrow \sum_i \lambda_i \psi_i(x) \psi_i(t)$ converges uniformly to a continuous function $g_x$

For any fixed $x \in \mathcal{X}$, we therefore have, for any $t \in \mathcal{X}$ (restricting the sum to the indices $i \geq 1$ such that $\lambda_i > 0$):

$$\sum_{i=m}^{m+\ell} \lambda_i \psi_i(x) \psi_i(t) \leq \left( \sum_{i=m}^{m+\ell} \lambda_i \psi_i(x)^2 \right)^{\frac{1}{2}} \left( \sum_{i=m}^{m+\ell} \lambda_i \psi_i(t)^2 \right)^{\frac{1}{2}}$$

$$\leq C_K \left( \sum_{i=m}^{m+\ell} \lambda_i \psi_i(x)^2 \right)^{\frac{1}{2}},$$

which tends to 0 uniformly in $t \in \mathcal{X}$. Therefore the series of functions $t \rightarrow \sum_i \lambda_i \psi_i(x) \psi_i(t)$ is uniformly Cauchy, continuous, and therefore converges uniformly to a continuous function $g_x$. 
Proof of Mercer’s Theorem (6/6)

\[ K_x = g_x \text{ in } L_2(\nu) \]

On the other hand, we can expand \( K_x \) over the ONB \( \{\psi_k, k \geq 1\} \) of \( L_2^2(\mathcal{X}) \):

\[
K_x = \sum_{k \geq 1} \langle K_x, \psi_k \rangle_{L_2^2(\mathcal{X})} \psi_k \\
= \sum_{k \geq 1} (L\psi_k)(x)\psi_k \\
= \sum_{k \geq 1} \lambda_k \psi_k(x)\psi_k \\
= \sum_{k \geq 1 : \lambda_k > 0} \lambda_k \psi_k(x)\psi_k,
\]

therefore \( K_x = g_x \text{ in } L_2(\nu) \), i.e., \( \|K_x - g_x\|_{L_2(\nu)} = 0 \).
Since $\nu$ is nondegenerate, and both $K_x$ and $g_x$ are continuous, this implies

$$\forall t \in \mathcal{X}, \quad K_x(t) = g_x(t) = \sum_{i} \lambda_i \psi_i(x) \psi_i(t),$$

and the convergence is uniform in $\mathcal{X} \times \mathcal{X}$ because $K$ is continuous.
Mercer kernels as inner products

Let $\ell^2$ denote the Hilbert space of real-valued sequences $u = (u_k)_{k \in \mathbb{N}}$ such that $\sum_{k \in \mathbb{N}} u_k^2 < +\infty$, endowed with the inner product $\langle u, v \rangle = \sum_{k \in \mathbb{N}} u_k v_k$.

**Corollary**

The mapping

$$\Phi : \mathcal{X} \mapsto \ell^2$$

$$\mathbf{x} \mapsto \left( \sqrt{\lambda_k} \psi_k (\mathbf{x}) \right)_{k \in \mathbb{N}}$$

is well defined, continuous, and satisfies

$$K (\mathbf{x}, \mathbf{t}) = \langle \Phi (\mathbf{x}), \Phi (\mathbf{t}) \rangle_{\ell^2}.$$
Proof of the corollary

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

- The continuity of \( \Phi \) results from:

\[
\| \Phi(x) - \Phi(t) \|_{\ell^2}^2 = \sum_{k=1}^{\infty} \lambda_k (\psi_k(x) - \psi_k(t))^2
= K(x, x) + K(t, t) - 2K(x, t)
\]
Summary

- This proof extends the proof valid when $\mathcal{X}$ is finite.
- This is a constructive proof, developed by Mercer (1905).
- The eigensystem ($\lambda_k$ and $\psi_k$) depend on the choice of the measure $d\nu(x)$: different $\nu$’s lead to different feature spaces for a given kernel and a given space $\mathcal{X}$
- Compactness and continuity are required. For instance, for $\mathcal{X} = \mathbb{R}^d$, the eigenvalues of:

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

are not necessarily countable, Mercer theorem does not hold. Other tools are thus required such as the Fourier transform for shift-invariant kernels.
Example 1: $[0, 1]$ (1/6)

- Consider the unit interval $\mathcal{X} = [0, 1]$ endowed with the Lebesgue measure $d\nu(x) = dx$
- Let a p.d. kernel on $\mathcal{X}$ of the form

$$K(x, t) = \kappa(x - t),$$

where $\kappa : \mathbb{R} \to \mathbb{R}$ is continuous and 1-periodic.
- To write Mercer’s expansion we need to find the eigenfunctions of $L_K$ by solving

$$(L_K \psi)(x) = \int_0^1 \kappa(x - t) \psi(t) \, dt = \lambda \psi(x)$$
Lemma

Let \((\psi_n)_{n \in \mathbb{N}}\) be the Fourier ONB of \(L^2([0, 1])\) given by \(\psi_0(x) = 1\) and

\[\forall n \geq 1, \quad \begin{cases} 
\psi_{2n-1}(x) = \sqrt{2} \sin(2\pi nx), \\
\psi_{2n}(x) = \sqrt{2} \cos(2\pi nx).
\end{cases}\]

Let the Fourier expansion of \(\kappa\) be\(^a\)

\[\forall x \in [0, 1], \quad \kappa(x) = \sum_{n=0}^{\infty} \hat{\kappa}_{2n} \psi_{2n}(x).\]

Then for any \(n \in \mathbb{N}\), \(\psi_n\) is an eigenfunction of \(L_K\) with eigenvalues \(\hat{\kappa}_0\) for \(\psi_0\) and \(\hat{\kappa}_{2n}/\sqrt{2}\) for \(\psi_{2n-1}\) and \(\psi_{2n}\).

\(^aK\) symmetric \(\implies\) \(\kappa\) even \(\implies\) \(\hat{\kappa}_{2n+1} = 0\) for \(n \in \mathbb{N}\).
Example 1: [0, 1] (3/6)

Proof sketch:

- \((\psi_n)_{n \in \mathbb{N}}\) is an ONB of \(L^2([0, 1])\) by direct computation of 
  \[
  \int_0^1 \psi_i(x) \psi_j(x) \, dx = \delta_{ij}.
  \]
- By trigonometric expansion of \(\sin(a + b)\) and \(\cos(a + b)\), show that
  \[
  \begin{align*}
  \psi_{2n}(x - t) &= \frac{1}{\sqrt{2}} \left[ \psi_{2n}(x) \psi_{2n}(t) + \psi_{2n-1}(x) \psi_{2n-1}(t) \right], \\
  \psi_{2n-1}(x - t) &= \frac{1}{\sqrt{2}} \left[ \psi_{2n-1}(x) \psi_{2n}(t) - \psi_{2n}(x) \psi_{2n-1}(t) \right].
  \end{align*}
  \]
- Then direct computation of \(L_K \psi_i\), e.g.,
  \[
  L_K \psi_{2n}(x) = \sum_{\ell=0}^{\infty} \hat{\kappa}_2 \ell \int_0^1 \psi_{2\ell}(x - t) \psi_{2n}(t) \, dt
  \]
  \[
  = \sum_{\ell=0}^{\infty} \frac{\hat{\kappa}_2 \ell}{\sqrt{2}} \int_0^1 \left[ \psi_{2\ell}(x) \psi_{2\ell}(t) + \psi_{2\ell-1}(x) \psi_{2\ell-1}(t) \right] \psi_{2n}(t) \, dt
  \]
  \[
  = \sum_{\ell=0}^{\infty} \frac{\hat{\kappa}_2 \ell}{\sqrt{2}} \psi_{2\ell}(x) \delta_{n \ell} = \frac{\hat{\kappa}_{2n}}{\sqrt{2}} \psi_{2n}(x). \quad \square
  \]
Example 1: $[0, 1]$ (4/6)

Remark: Mercer’s theorem is obviously correct. All $\psi_k$’s are continuous, and for any $x, t \in [0, 1]$ the Mercer expansion of the kernel is:

$$
K(x, t) = \hat{\kappa}_0 + \sum_{n=1}^{\infty} \frac{\hat{\kappa}_{2n}}{\sqrt{2}} \left[ \psi_{2n-1}(x)\psi_{2n-1}(t) + \psi_{2n}(x)\psi_{2n}(t) \right]
$$

$$
= \sum_{n=0}^{\infty} \hat{\kappa}_{2n}\psi_{2n}(x - t)
$$

$$
= \kappa(x - t),
$$

with absolute and uniform convergence (because $\kappa$ is continuous).
Example 1: [0, 1] (5/6)

Example: polynomial decay of eigenvalues

For any $\beta \in \mathbb{N}^*$, let

$$
\begin{align*}
\hat{\kappa}_0 &= 0, \\
\hat{\kappa}_{2n} &= \sqrt{2} n^{-2\beta} \text{ for } n \geq 1.
\end{align*}
$$

Then the corresponding kernel is

$$
\forall x, t \in [0, 1], \quad K(x, t) = \frac{1}{(2\beta)!} B_{2\beta}(x - t - \lfloor x - t \rfloor),
$$

where $B_{2\beta}$ is the $(2\beta)$-th Bernoulli polynomial\(^a\), e.g.,

$$
B_2(x) = x^2 - x + 1/6, \quad B_4(x) = x^4 - 2x^3 + x^2 - 1/30, \ldots
$$

\(^a\)https://en.wikipedia.org/wiki/Bernoulli_polynomials

*Proof left as exercise (check Fourier expansion of Bernoulli polynomials).*
Example 1: [0, 1] (6/6)

Example: exponential decay of eigenvalues

For any $\rho \in \mathbb{R}_+$, let

\[
\begin{align*}
  \hat{\kappa}_0 &= 0, \\
  \hat{\kappa}_{2n} &= e^{-\rho n} \text{ for } n \geq 1.
\end{align*}
\]

Then the corresponding kernel is

\[
K(x, t) = \frac{\sqrt{2}e^{\rho} \cos(2\pi(x-t)) - 1}{e^{2\rho} - 2e^{\rho} \cos(2\pi(x-t)) + 1}.
\]

Proof left as exercise (or check Bach, 2013, p.21).
Example 2: $S^{d-1}$ (1/6)

Consider the unit sphere in $\mathbb{R}^d$:

$$\mathcal{X} = S^{d-1} = \left\{ x \in \mathbb{R}^d : \|x\| = 1 \right\}$$

Let $\nu$ be the Lebesgue measure on $S^{d-1}$. Note that:

$$\nu(S^{d-1}) = \frac{2\pi^{d/2}}{\Gamma\left(\frac{d}{2}\right)}$$
Example 2: \( S^{d-1} \) (2/6)

- Let a p.d. kernel on \( S^{d-1} \) of the form:

\[
K(x, t) = \varphi(x^\top t),
\]

where \( \varphi: [-1, 1] \to \mathbb{R} \) is continuous.

- To write Mercer’s expansion we need to find the eigenfunctions by solving

\[
\int_{S^{d-1}} \varphi(x^\top t) \psi(t) d\nu(t) = \lambda \psi(x)
\]

- For that purpose study polynomials that solve the Laplace equation:

\[
\Delta f = \frac{\partial^2 f}{\partial x_1^2} + \ldots + \frac{\partial^2 f}{\partial x_d^2} = 0
\]

where \( \Delta \) is the Laplacian operator on \( \mathbb{R}^d \).
Example 2: $S^{d-1}$ (3/6)

Definition (Spherical harmonics)

- A homogeneous polynomial of degree $k \geq 0$ in $\mathbb{R}^d$ whose Laplacian vanishes is called a homogeneous harmonic of order $k$.
- A spherical harmonic of order $k$ is a homogeneous harmonic of order $k$ on the unit sphere $S^{d-1}$.

The set $\mathcal{Y}_k(d)$ of spherical harmonics is a vector space of dimension

$$N(n, k) = \text{dim} (\mathcal{Y}_k(d)) = \frac{(2k + d - 2)(k + d - 3)!}{k!(d - 2)!}.$$
Example 2: \( S^{d-1} (4/6) \)

Spherical harmonics form the Mercer’s eigenfunctions, because:

**Theorem (Funk-Hecke) (e.g., Müller, 1998, p.30)**

For any \( x \in S^{d-1} \), \( Y_k \in \mathcal{Y}_k(d) \) and \( \varphi \in C([-1,1]) \),

\[
\int_{S^{d-1}} \varphi \left( x^\top t \right) Y_k(t) \, d\nu(t) = \lambda_k Y_k(x)
\]

where

\[
\lambda_k = \nu \left( S^{d-2} \right) \int_{-1}^{1} \varphi(t) P_k(d; t)(1 - t^2)^{\frac{d-3}{2}} \, dt
\]

and \( P_k(d; t) \) is the Legendre polynomial of degree \( k \) in dimension \( d \).

When \( \varphi \in C^k([-1,1]) \) we have Rodrigues rule (Müller, 1998, p.23):

\[
\lambda_k = \nu \left( S^{d-2} \right) \frac{\Gamma \left( \frac{d-1}{2} \right)}{2^k \Gamma \left( k + \frac{d-1}{2} \right)} \int_{-1}^{1} \varphi^{(k)}(t) (1 - t^2)^{k+\frac{d-3}{2}} \, dt
\]
Example 2: $S^{d-1}$ (5/6)

- For any $k \geq 0$, let $\{ Y_{k,j}(d; x) \}_{j=1}^{N(d;k)}$ an orthonormal basis of $\mathcal{Y}_k(d)$.
- Spherical harmonics $\left\{ \left\{ Y_{k,j}(d; x) \right\}_{j=1}^{N(d;k)} \right\}_{k=0}^{\infty}$ form an orthonormal basis for $L^2(S^{d-1})$.
- Therefore, for any kernel $K(x, t) = \varphi(x^\top t)$ on $S^{d-1}$ the Mercer eigenvalues are exactly the $\lambda_k$'s, with corresponding orthonormal eigenfunctions $\{ Y_{k,j}(d; x) \}_{j=1}^{N(d;k)}$.
- Note that eigenfunctions are the same for different $\varphi$'s, only the eigenvalues change.
Example 2: $S^{d-1}$ (6/6)

- Take $d = 2$ and $K(x, t) = (1 + x^\top t)^2$ for $x, t \in S^1$
- Using Rodrigeus rule we get 3 nonzero eigenvalues:
  \[
  \lambda_0 = 3\pi, \quad \lambda_1 = 2\pi, \quad \lambda_2 = \frac{\pi}{2}
  \]
  with multiplicities 1, 2 and 2
- Corresponding eigenfunctions:
  \[
  \left( \frac{1}{\sqrt{2\pi}}, \frac{x_1}{\sqrt{\pi}}, \frac{x_2}{\sqrt{\pi}}, \frac{x_1x_2}{\sqrt{\pi}}, \frac{x_1^2 - x_2^2}{\sqrt{\pi}} \right)
  \]
- The resulting Mercer feature map is
  \[
  \Phi(x) = \left( \sqrt{\frac{3}{2}}, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, \frac{x_1^2 - x_2^2}{\sqrt{2}} \right)
  \]
- Obviously, $\Phi(x)^\top \Phi(t) = K(x, t)$ for $x, t \in S^1$ (exercise)
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 $L^2_\nu(X)$ defined by:

$$\forall f \in L^2_\nu(X), (L_K f)(x) = \int K(x, t) f(t) \, d\nu(t).$$

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

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

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

RKHS construction

**Theorem**

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

\[ \mathcal{H} = \left\{ f = \sum_{i=1}^{\infty} a_i \psi_i, \quad \text{with} \quad \sum_{k=1}^{\infty} \frac{a_k^2}{\lambda_k} < \infty \right\} \]

endowed with the inner product:

\[ \langle f, g \rangle_{\mathcal{H}} = \sum_{k=1}^{\infty} \frac{a_k b_k}{\lambda_k}, \quad \text{for} \quad f = \sum_{k} a_k \psi_k, \ g = \sum_{k} b_k \psi_k. \]

**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.
Sketch

In order to show that $\mathcal{H}$ 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 $x \in \mathcal{X}$, $K_x \in \mathcal{H}$,
3. for any $x \in \mathcal{X}$ and $f \in \mathcal{H}$, $f(x) = \langle f, K_x \rangle_{\mathcal{H}}$. 
\[ L^\frac{1}{2}_K : L^2_\nu(X) \to \mathcal{H} \]

Indeed the function:

\[
\sum_{i=1}^{\infty} a_i \psi_i \mapsto \sum_{i=1}^{\infty} a_i \sqrt{\lambda_i} \psi_i
\]

is an isomorphism, therefore \( \mathcal{H} \) is a Hilbert space, like \( L^2_\nu(X) \). \( \square \)
Proof (3/6)

\( \mathcal{H} \) is a space of continuous functions

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

\[
\left| f(x) \right| = \left| \sum_{i=1}^{\infty} a_i \psi_i(x) \right| = \left| \sum_{i=1}^{\infty} \frac{a_i}{\sqrt{\lambda_i}} \sqrt{\lambda_i} \psi_i(x) \right|
\]

\[
\leq \left( \sum_{i=1}^{\infty} \frac{a_i^2}{\lambda_i} \right)^{\frac{1}{2}} \cdot \left( \sum_{i=1}^{\infty} \lambda_i \psi_i(x)^2 \right)^{\frac{1}{2}}
\]

\[
= \| f \|_{\mathcal{H} K(x, x)}^{\frac{1}{2}}
\]

\[
= \| f \|_{\mathcal{H}} \sqrt{C_K}.
\]

Therefore, convergence in \( \| . \|_{\mathcal{H}} \) implies uniform convergence for functions.
$\mathcal{H}$ is a space of continuous functions (cont.)

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

Let $f_c$ the continuous limit function. Then $f_c \in L^2_\nu(X)$ and

$$\| f_n - f_c \|_{L^2_\nu(X)} \rightarrow 0,$$

On the other hand,

$$\| f - f_n \|_{L^2_\nu(X)} \leq \lambda_1 \| f - f_n \|_{\mathcal{H}} \rightarrow 0,$$

therefore $f = f_c$. □
Proof (5/6)

\[ K_x \in \mathcal{H} \]

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

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

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

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

therefore \( \varphi_x = K_x \in \mathcal{H} \). \( \square \)
**Proof (6/6)**

\[ f(x) = \langle f, K_x \rangle_{\mathcal{H}} \]

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

\[ K_x = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i , \]

therefore:

\[ \langle f, K_x \rangle_{\mathcal{H}} = \sum_{i=1}^{\infty} \frac{\lambda_i \psi_i(x) a_i}{\lambda_i} = \sum_{i=1}^{\infty} a_i \psi_i(x) = f(x) , \]

which concludes the proof. \( \square \)
Remarks

- Although $\mathcal{H}$ was built from the eigenfunctions of $L_K$, which depend on the choice of the measure $d\nu(x)$, we know by uniqueness of the RKHS that $\mathcal{H}$ is independent 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:

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

  The RKHS is a well-defined ellipsoid with axes given by the eigenfunctions.
Example: Sobolev space of periodic functions on $[0, 1]$

**Corollary**

For $\beta \in \mathbb{N}_*$, let the Mercer kernel with polynomially decaying eigenvalues:

$$
\forall x, t \in [0, 1], \quad K(x, t) = \frac{1}{(2\beta)!} B_{2\beta}(x - t - \lfloor x - t \rfloor),
$$

where $B_{2\beta}$ is the $(2\beta)$-th Bernoulli polynomial. Then the RKHS is the set of functions $f : [0, 1] \rightarrow \mathbb{R}$ whose Fourier coefficients satisfy:

$$
\| f \|_{\mathcal{H}}^2 := \sum_{n=1}^{\infty} \left( \hat{f}_{2n-1}^2 + \hat{f}_{2n}^2 \right) n^{2\beta} < +\infty.
$$

This is the **Sobolev space** of functions $f$ such that $f^{(i)}$ is absolutely continuous and $f^{(i)}(0) = f^{(i)}(1)$, for $i = 0, \ldots, \beta - 1$, and

$$
\| f \|_{\mathcal{H}}^2 = \pi^{-2\beta} \int_0^1 \left( f^{(\beta)}(x) \right)^2 dx.
$$
Proof sketch

- The characterization of the RKHS in terms of Fourier coefficients is a direct application of the previous result, noting that the Fourier basis is an ONB of eigenfunctions of $L_K$, and that the corresponding eigenvalues are $n^{-2\beta}$.

- For the characterization as a Sobolev space, we use Parceval equality to rewrite the Sobolev norm as the $\ell_2$ norm of the Fourier coefficients of $f^{(\beta)}$, which are (roughly) the Fourier coefficients of $f$ multiplied by $n^{\beta}$. For details, see Tsybakov (2004, Proposition 1.14).
The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
  - Green kernels
  - Mercer kernels
  - Convergence rates of KRR for Mercer kernels
    - Shift-invariant kernels
    - Generalization to semigroups
- Kernels for probabilistic models
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs
Isomorphism between $\mathcal{H}$ and $L^2_\nu(\mathcal{X})$

- We saw that

$$L^\frac{1}{2}_K : L^2_\nu(\mathcal{X}) \to \mathcal{H}$$

$$\sum_{i=1}^{\infty} a_i \psi_i \mapsto \sum_{i=1}^{\infty} a_i \sqrt{\lambda_i} \psi_i$$

is an isomorphism between $\mathcal{H}$ and $L^2_\nu(\mathcal{X})$, i.e.,

$$\forall f \in L^2_\nu(\mathcal{X}) \ , \quad \| f \|_{L^2_\nu(\mathcal{X})} = \| L^\frac{1}{2}_K f \|_{\mathcal{H}} ,$$

and conversely,

$$\forall f \in \mathcal{H} , \quad \| f \|_{\mathcal{H}} = \| L^{-\frac{1}{2}}_K f \|_{L^2_\nu(\mathcal{X})} .$$

- This can be useful to compute $L^2_\nu(\mathcal{X})$ norms using RKHS theory, e.g., to study the performance of kernel ridge regression (KRR).
Remember KRR

Given \((x_1, \ldots, x_n) \in \mathcal{X}^n\) and \((y_1, \ldots, y_n) \in \mathbb{R}^n\), KRR solves for any \(\lambda > 0\):

\[
\hat{f}_\lambda = \arg \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \|f\|_H^2.
\]

The solution is

\[
\hat{f}_\lambda(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x), \quad \text{where} \quad \alpha = (K + \lambda n I)^{-1} y.
\]
Let $K$ be a Mercer kernel over the compact set $\mathcal{X}$ and nondegenerate probability measure $\nu$ (i.e., $\nu(\mathcal{X}) = 1$). Let $\lambda_1 \geq \lambda_2 \geq ... \geq 0$ be the eigenvalues of $L_K$, $\{\psi_i, i \geq 1\}$ the eigenvectors, and $\{\varphi_i = \sqrt{\lambda_i} \psi_i, i \geq 1\}$ an ONB of $\mathcal{H}$.

Let $(X, Y)$ be random variables with distribution $P$, such that

$$X \in \mathcal{X} \text{ has distribution } \nu$$

and

$$Y = f^*(X) + \epsilon \text{ where } f^* \in \mathcal{H} \text{ and } \epsilon \sim \mathcal{N}(0, \sigma^2).$$

We assume $(x_i, y_i)_{i=1,...,n}$ are i.i.d. realizations of $(X, Y)$.

We want to estimate the performance of KRR in terms of mean squared error:

$$\text{MSE}(\hat{f}_\lambda) = \mathbb{E}(Y - \hat{f}_\lambda(X))^2.$$
Decomposition of the MSE

Lemma

Let $\beta^* \in \ell^2$ such that $f^* = \sum_{i \geq 1} \beta^*_i \varphi_i$, let $\Phi_N$ the $n \times \infty$ matrix given by $\Phi_n = (\varphi_j(x_i))_{1 \leq i \leq n; 1 \leq j < +\infty}$. and $T : \ell_2 \rightarrow \ell_2$ be the diagonal operator $T(a_1, a_2, \ldots) = (\lambda_1 a_1, \lambda_2 a_2, \ldots)$.

Then it holds

$$\text{MSE}(\hat{f}_\lambda) - \text{MSE}(f^*) = B_\lambda + V_\lambda,$$

where

$$B_\lambda = \mathbb{E} \| T^{\frac{1}{2}} \left( I - (\Phi_n^\top \Phi_n + \lambda n I)^{-1} \Phi_n^\top \Phi_n \right) \beta^* \|_{\ell^2}^2,$$

$$V_\lambda = \mathbb{E} \| T^{\frac{1}{2}} \left( \Phi_n^\top \Phi_n + \lambda n I \right)^{-1} \Phi_n^\top \varepsilon \|_{\ell^2}^2.$$

This corresponds to a classical decomposition of excess MSE as "bias + variance". Note that $B_\lambda$ increases with $\lambda$, but $V_\lambda$ decreases with $\lambda$. 
Decomposition of the MSE: Proof (1/5)

Since $\epsilon$ is independent of $X$ and $\hat{f}_\lambda$, and $\mathbb{E}\epsilon = 0$ we have

$$MSE(\hat{f}_\lambda) = \mathbb{E} \left( f^*(X) - \hat{f}_\lambda(X) + \epsilon \right)^2$$

$$= \mathbb{E} \left( f^*(X) - \hat{f}_\lambda(X) \right)^2 + \mathbb{E}\epsilon^2$$

$$= \mathbb{E}\| f^* - \hat{f}_\lambda \|_{L^2_\nu(X)}^2 + MSE(f^*).$$

Using the isometry between $L^2_\nu(X)$ and $\mathcal{H}$, we obtain

$$MSE(\hat{f}_\lambda) - MSE(f^*) = \mathbb{E}\| L^{\frac{1}{2}}_K(f^* - \hat{f}_\lambda) \|_{\mathcal{H}}^2.$$
Decomposition of the MSE: Proof (2/5)

- \( \{ \varphi_i = \sqrt{\lambda_i} \psi_i \} , i \geq 1 \) is an ONB of \( \mathcal{H} \), we can define the linear isomorphism:

\[
e : \mathcal{H} \to \ell^2
\]

\[
f = \sum_{i \geq 1} a_i \varphi_i \mapsto (a_1, a_2 \ldots)^\top
\]

- In other words,

\[
e(f)_i = \langle f, \varphi_i \rangle_{\mathcal{H}}.
\]

- In particular, for any \( x \in \mathcal{X} \),

\[
e(K_x) = (\varphi_1(x), \varphi_2(x), \ldots)^\top.
\]

- In that base \( L_K \) is a diagonal operator \( T = \text{diag}(\lambda_1, \lambda_2, \ldots) \), i.e.,

\[
\forall f = \sum_{i \geq 1} a_i \varphi_i \in \mathcal{H} , \quad e(L_K f) = T e(f) = (\lambda_1 a_1, \lambda_2 a_2, \ldots)^\top.
\]
Decomposition of the MSE: Proof (3/5)

- Let $\Phi_n = (e(K_{x_1}), \ldots, e(K_{x_n}))^\top$, i.e.,
  $$
  \Phi_n = (\varphi_j(x_i))_{1 \leq i \leq n; 1 \leq j < +\infty}.
  $$

- Then $\hat{f}_\lambda = \sum_{i=1}^n \alpha_i K_{x_i}$ translates to
  $$
  e(\hat{f}_\lambda) = \sum_{i=1}^n \alpha_i e(K_{x_i}) = \Phi_n^\top \alpha.
  $$

- Notice that
  $$
  [\Phi_n \Phi_n^\top]_{ij} = \langle e(K_{x_i}), e(K_{x_j}) \rangle_{\ell^2} = \langle K_{x_i}, K_{x_j} \rangle_{H} = K(x_i, x_j),
  $$
  so $\Phi_n \Phi_n^\top = K$ and $\alpha = (K + \lambda n I)^{-1} y$ translates to
  $$
  \alpha = \left(\Phi_n \Phi_n^\top + \lambda n I\right)^{-1} y.
  $$

- Putting it all together, and using the matrix inversion lemma:
  $$
  e(\hat{f}_\lambda) = \Phi_n^\top \left(\Phi_n \Phi_n^\top + \lambda n I\right)^{-1} y = \left(\Phi_n^\top \Phi_n + \lambda n I\right)^{-1} \Phi_n^\top y.
  $$
Decomposition of the MSE: Proof (4/5)

- Let $\beta^* = (\beta_1^*, \beta_2^*, \ldots)^\top = e(f^*)$, i.e.,
  \[ f^* = \sum_{i \geq 1} \beta_i^* \varphi_i. \]

In particular, for any $x \in \mathcal{X}$,
\[ f^*(x) = \langle f^*, K_x \rangle_{\mathcal{H}} = \langle \beta^*, e(K_x) \rangle_{\ell^2}. \]

- Then $y_i = f^*(x_i) + \epsilon_i$ for $i = 1, \ldots, n$ translates to
  \[ y = \Phi_n \beta^* + \epsilon, \]
where $\epsilon = (\epsilon_1, \ldots, \epsilon_n)^\top$.  

Decomposition of the MSE: Proof (5/5)

This gives

\[ e(f^* - \hat{f}_\lambda) = \beta^* - \left( \Phi_n^\top \Phi_n + \lambda n\mathbf{I} \right)^{-1} \Phi_n^\top (\Phi_n \beta^* + \epsilon) \]

\[ = \left( \mathbf{I} - \left( \Phi_n^\top \Phi_n + \lambda n\mathbf{I} \right)^{-1} \Phi_n^\top \Phi_n \right) \beta^* - \left( \Phi_n^\top \Phi_n + \lambda n\mathbf{I} \right)^{-1} \Phi_n^\top \epsilon, \]

and therefore, since \( \epsilon \) is independent of \( \Phi_n \):

\[ \mathbb{E} \| L_{\mathcal{K}}^{\frac{1}{2}} (f^* - \hat{f}_\lambda) \|_\mathcal{H}^2 = \mathbb{E} \| e \left( L_{\mathcal{K}}^{\frac{1}{2}} (f^* - \hat{f}_\lambda) \right) \|_\ell^2^2 \]

\[ = \mathbb{E} \| T^{\frac{1}{2}} e (f^* - \hat{f}_\lambda) \|_\ell^2^2 \]

\[ = \mathbb{E} \| T^{\frac{1}{2}} \left( \mathbf{I} - \left( \Phi_n^\top \Phi_n + \lambda n\mathbf{I} \right)^{-1} \Phi_n^\top \Phi_n \right) \beta^* \|_\ell^2^2 \]

\[ + \mathbb{E} \| T^{\frac{1}{2}} \left( \Phi_n^\top \Phi_n + \lambda n\mathbf{I} \right)^{-1} \Phi_n^\top \epsilon \|_\ell^2^2. \]
B_\lambda and V_\lambda depend on the data through \Phi_n\Phi_n^\top, which is a random operator \ell^2 \rightarrow \ell^2.

For "large n", we note that, for any i, j \geq 1:

\[
\left[\Phi_n\Phi_n^\top\right]_{ij} = \sum_{k=1}^{n} \varphi_i(x_k)\varphi_j(x_k) \approx n\langle \varphi_i, \varphi_j\rangle_{L_2^2(\mathcal{X})} = n\sqrt{\mu_i\mu_j}\delta_{ij},
\]

so

\Phi_n\Phi_n^\top \approx nT.

We now study B_\lambda and V_\lambda under the approximation "\Phi_n\Phi_n^\top = nT" (and call \tilde{B}_\lambda and \tilde{V}_\lambda the corresponding approximations).

The difference between B_\lambda and \tilde{B}_\lambda (resp. V_\lambda and \tilde{V}_\lambda) can be studied rigorously but will not change much the main results we will get; see, e.g., Dicker et al. (2015) for details.
Upper bounds on the bias and variance

**Theorem**

*For any* $J \geq 1$,

$$\tilde{B}_\lambda \leq \left(\frac{\lambda^2}{\lambda_J} + \lambda_{J+1}\right) \|f^*\|_{\mathcal{H}}^2,$$

*and*

$$\tilde{V}_\lambda \leq \frac{\sigma^2}{n} \left[ J + \sum_{i=J+1}^{+\infty} \frac{\lambda_i}{4\lambda} \right].$$

The integer $J$ (and $\lambda$) will be optimized later, depending on the assumptions we make on $f^*$ and on the decrease of $\lambda_i$. 
Proof: bias (1/2)

- Using $T = \text{diag}(\lambda_i; i \geq 1)$ and $\Phi_n \Phi_n^\top = nT$, we get

$$T^{\frac{1}{2}} \left( I - \left( \Phi_n^\top \Phi_n + \lambda n I \right)^{-1} \Phi_n^\top \Phi_n \right) = \text{diag} \left( \frac{\lambda \sqrt{\lambda_i}}{\lambda + \lambda_i}; i \geq 1 \right),$$

and therefore, for any $J \geq 1$:

$$\tilde{B}_\lambda = \sum_{i=1}^{J} \frac{\lambda^2 \lambda_i}{(\lambda + \lambda_i)^2} (\beta_i^*)^2 + \sum_{i \geq J+1}^{\infty} \frac{\lambda^2 \lambda_i}{(\lambda + \lambda_i)^2} (\beta_i^*)^2.$$

- For the first term, we use the fact that $\frac{\lambda_i^2}{(\lambda + \lambda_i)^2} \leq 1$, and that $\lambda_i \geq \lambda_j$ for $i \leq J$, to get

$$\sum_{i=1}^{J} \frac{\lambda^2 \lambda_i}{(\lambda + \lambda_i)^2} (\beta_i^*)^2 = \sum_{i=1}^{J} \frac{\lambda^2}{\lambda_i} \frac{\lambda_i^2}{(\lambda + \lambda_i)^2} (\beta_i^*)^2$$

$$\leq \frac{\lambda^2}{\lambda J} \sum_{i=1}^{J} (\beta_i^*)^2 \leq \frac{\lambda^2}{\lambda J} \| \beta^* \|_{\ell^2}^2.$$
Proof: bias (2/2)

- For the second term, we use the fact that \( \frac{\lambda^2}{(\lambda + \lambda_i)^2} \leq 1 \), and that \( \lambda_i \leq \lambda_{J+1} \) for \( i \geq J + 1 \), to get

\[
\sum_{i \geq J+1} \frac{\lambda^2 \lambda_i}{(\lambda + \lambda_i)^2} (\beta_i^*)^2 \leq \lambda_{J+1} \sum_{i \geq J+1} (\beta_i^*)^2 \leq \lambda_{J+1} \| \beta^* \|^2_{\ell^2}.
\]

- Noting that \( \| \beta^* \|_{\ell^2} = \| f^* \|_{\mathcal{H}} \), we finally get

\[
\tilde{B}_\lambda \leq \left( \frac{\lambda^2}{\lambda_j} + \lambda_{J+1} \right) \| f^* \|^2_{\mathcal{H}} .
\]
Proof: variance \((1/2)\)

Using \(T = \text{diag}(\lambda_i; i \geq 1), \Phi_n \Phi_n^\top = nT\) and \(\mathbb{E}\epsilon\epsilon^\top = \sigma^2 I\), we get

\[
\hat{V}_\lambda = \mathbb{E}\| T^{1/2} \left( \Phi_n^\top \Phi_n + \lambda n I \right)^{-1} \Phi_n \epsilon \|_{\ell^2}^2 \\
= \frac{1}{n^2} \mathbb{E}\| T^{1/2} (T + \lambda I)^{-1} \Phi_n \epsilon \|_{\ell^2}^2 \\
= \frac{1}{n^2} \mathbb{E}\text{Trace} \left[ T^{1/2} (T + \lambda I)^{-1} \Phi_n \epsilon\epsilon^\top \Phi_n (T + \lambda I)^{-1} T^{1/2} \right] \\
= \frac{1}{n^2} \text{Trace} \left[ T^{1/2} (T + \lambda I)^{-1} \Phi_n \mathbb{E} \left( \epsilon\epsilon^\top \right) \Phi_n (T + \lambda I)^{-1} T^{1/2} \right] \\
= \frac{\sigma^2}{n} \text{Trace} \left[ T^{1/2} (T + \lambda I)^{-1} T (T + \lambda I)^{-1} T^{1/2} \right] \\
= \frac{\sigma^2}{n} \left[ \sum_{i=1}^{J} \frac{\lambda_i^2}{(\lambda_i + \lambda)^2} + \sum_{i=J+1}^{+\infty} \frac{\lambda_i^2}{(\lambda_i + \lambda)^2} \right].
\]
Proof: variance (2/2)

- For the first term, we just use \( \frac{\lambda_i^2}{(\lambda_i + \lambda)^2} \leq 1 \) to get

\[
\sum_{i=1}^{J} \frac{\lambda_i^2}{(\lambda_i + \lambda)^2} \leq J.
\]

- For the second term, we use the fact that \( t \to \frac{t}{(t+\lambda)^2} \) reaches its maximum at \( t = \lambda \) equal to \( \frac{1}{4\lambda} \), therefore

\[
\sum_{i=J+1}^{+\infty} \frac{\lambda_i^2}{(\lambda_i + \lambda)^2} \leq \sum_{i=J+1}^{+\infty} \frac{\lambda_i}{4\lambda}.
\]

- Combining both terms finally gives

\[
\tilde{\nu}_\lambda \leq \frac{\sigma^2}{n} \left[ J + \sum_{i=J+1}^{+\infty} \frac{\lambda_i}{4\lambda} \right].
\]
Corollary: rates of convergence of KRR

- **Polynomial-decay kernels.** Suppose there are constants $C > 0$ and $s > 1$ such that $0 < \lambda_i \leq Ci^{-s}$ for $i = 1, 2, \ldots$. Let $\lambda = n^{-\frac{s}{s+1}}$. Then

\[ \tilde{B}_\lambda + \tilde{V}_\lambda \leq O\left\{ (\| f^* \|_{\mathcal{H}}^2 + \sigma^2) n^{-\frac{s}{s+1}} \right\}. \]

- **Exponential-decay kernels.** Suppose there are constants $C > 0$ and $\alpha > 0$ such that $0 < \lambda_i \leq Ce^{-\alpha i}$ for $i = 1, 2, \ldots$. Let $\lambda = n^{-1} \log(n)$. Then

\[ \tilde{B}_\lambda + \tilde{V}_\lambda \leq O\left\{ (\| f^* \|_{\mathcal{H}}^2 + \sigma^2) \frac{\log(n)}{n} \right\}. \]

- **Finite rank kernels.** Suppose there is $J \geq 1$ such that $\lambda_J = \lambda_{J+1} = \ldots = 0$. Let $\lambda = n^{-1}$. Then

\[ \tilde{B}_\lambda + \tilde{V}_\lambda \leq O\left\{ (\| f^* \|_{\mathcal{H}}^2 + \sigma^2) \frac{J}{n} \right\}. \]
Remarks

The same result holds for $B_\lambda + V_\lambda$, see Dicker et al. (2015, corollary 1-4). We follow and adapt their proof.

The constants in the "big-O" notation only depend on the kernel $K$ and the measure $d\nu(x)$.

The rates are minimax optimal (Caponnetto and De Vito, 2007).

In particular, for polynomial-decay kernels, $B_H(r) \subset L_{\nu}^2(\mathcal{X})$ is a Sobolev space of $q-1$ times absolutely continuous and differentiable functions $f$ with $\| f^q \|_{L_{\nu}^2(\mathcal{X})} < +\infty$, for $s = 2q$. We recover the standard optimal convergence rate of nonparametric regression $n^{-\frac{2q}{2q+1}}$ (Tsybakov, 2004).

If we make additional assumptions on $f^*$, e.g., not only $\sum_{i \geq 1} (\beta_i^*)^2$ but also $\sum_{i \geq 1} i^{\tau} (\beta_i^*)^2$ for $\tau > 0$, or $\beta_i^* = 0$ for $i > J$, then we can get faster convergence rate, which are also minimax optimal for the class of functions considered. We say that KRR is adaptive (Caponnetto and De Vito, 2007; Dicker et al., 2015).
Proof for polynomial-decay kernels (1/3)

Let \( J \) such that \( \lambda_{J+1} \leq \lambda \leq \lambda_J \).

For the bias, we immediately get

\[
\frac{\lambda^2}{\lambda_J} \leq \lambda \quad \text{and} \quad \lambda_{J+1} \leq \lambda,
\]

therefore

\[
\tilde{B}_\lambda \leq 2\lambda \|f^*\|_{\mathcal{H}}^2 = 2n^{-\frac{s}{s+1}} \|f^*\|_{\mathcal{H}}^2.
\]

For the variance, we need to upper bound \( J \) and \( \sum_{i \geq J+1} \lambda_i \).

\( \lambda \leq \lambda_J \leq CJ^{-s} \), therefore

\[
J \leq C^{\frac{1}{s}} \lambda^{-\frac{1}{s}} = C^{\frac{1}{s}} n^{\frac{1}{s+1}}.
\]
To upper bound the sum, let $J_0 = \lceil C^{\frac{1}{s}} n^{\frac{1}{s+1}} \rceil + 1$. Then:

$$
\sum_{i=J+1}^{+\infty} \lambda_i = \sum_{i=J+1}^{J_0} \lambda_i + \sum_{i=J_0+1}^{+\infty} \lambda_i
\leq J_0 \lambda + C \int_{J_0}^{+\infty} t^{-s} dt
\leq J_0 n^{-\frac{s}{s+1}} + \frac{C}{s-1} J_0^{1-s}.
$$

Since $J_0 \leq C^{\frac{1}{s}} n^{\frac{1}{s+1}} + 1$ and $1 \leq n^{\frac{1}{s+1}}$,

$$
J_0 n^{-\frac{s}{s+1}} \leq \left( C^{\frac{1}{s}} + 1 \right) n^{\frac{1-s}{s+1}}.
$$

Since $J_0 \geq C^{\frac{1}{s}} n^{\frac{1}{s+1}}$,

$$
\frac{C}{s-1} J_0^{1-s} \leq \frac{C^{\frac{1}{s}} n^{\frac{1-s}{s+1}}}{s-1}.
$$
Proof for polynomial-decay kernels (3/3)

Therefore the sum is upper bounded by

\[
\sum_{i=J+1}^{+\infty} \lambda_i \leq \left( \frac{s}{s-1} C_{1s}^{\frac{1}{s}} + 1 \right) n^{\frac{1-s}{s+1}}.
\]

Finally,

\[
\tilde{V}_{\lambda} = \frac{\sigma^2}{n} \left[ J + \frac{1}{4} \sum_{i=J+1}^{+\infty} \lambda_i n^{-\frac{s}{s+1}} \right]
\]

\[
\leq \frac{\sigma^2}{n} \left[ C_{1s}^{\frac{1}{s}} n^{\frac{1}{s+1}} + \frac{1}{4} \left( \frac{s}{s-1} C_{1s}^{\frac{1}{s}} + 1 \right) n^{\frac{1-s}{s+1}} n^{-\frac{s}{s+1}} \right]
\]

\[
\leq \sigma^2 \left[ C_{1s}^{\frac{1}{s}} \left( 1 + \frac{s}{4(s-1)} \right) + \frac{1}{4} \right] n^{-\frac{s}{s+1}}. \quad \square
\]
Proof sketch for exponential-decay kernels

- We proceed similarly.
- From $\lambda \leq \lambda_J$ we deduce $J \leq O(\log(n))$.
- Using $J_0 = \lceil \alpha^{-1} \log(n) \rceil + 1$ we deduce $\sum_{i \geq J+1} \lambda_i \leq 0 \left( \frac{\log(n)^2}{n} \right)$.
- Details left as exercise; see Dicker et al. (2015, corollary 2). ☐
Proof for finite-rank kernels

- We use a simpler upper bound on $\tilde{B}_\lambda$: using the fact that
  \[ \frac{t}{(t+\lambda)^2} \leq \frac{1}{4\lambda} \]
  for any $t$, and $\lambda_i = 0$ for $i > J$:

  \[ \tilde{B}_\lambda \leq \frac{\lambda}{4} \| f^* \|_{\mathcal{H}}^2. \]

- For the variance, our bound simplifies to

  \[ \tilde{V}_\lambda \leq \frac{\sigma^2 J}{n}. \]

- Taking $\lambda = n^{-1}$ and summing this inequalities gives the result. □
The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
  - Green kernels
  - Mercer kernels
  - Convergence rates of KRR for Mercer kernels
  - Shift-invariant kernels
    - Generalization to semigroups
- Kernels for probabilistic models
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs
Motivation

- Let us suppose that $\mathcal{X}$ is not compact, for example $\mathcal{X} = \mathbb{R}^d$.
- In that case, the eigenvalues of:

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

are not necessarily countable, Mercer theorem does not hold.
- Fourier transforms provide a convenient extension for translation invariant kernels, i.e., kernels of the form $K(x, y) = \varphi(x - y)$.
- Harmonic analysis also bring kernels well beyond vector spaces, e.g., groups and semigroups.
Translation invariant kernels on \( \mathbb{Z} \)

**Definition**

A kernel \( K : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R} \) is called **translation invariant** (t.i.), or **shift-invariant**, if it only depends on the difference between its argument, i.e.:

\[
\forall x, y \in \mathbb{Z}, \quad K(x, y) = a_{x-y}
\]

for some sequence \( \{a_n\}_{n \in \mathbb{Z}} \). Such a sequence is called positive definite if the corresponding kernel \( K \) is p.d.
Translation invariant kernels on $\mathbb{Z}$

**Definition**

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

$$\forall x, y \in \mathbb{Z}, \quad K(x, y) = a_{x-y}$$

for some sequence $\{a_n\}_{n \in \mathbb{Z}}$. Such a sequence is called **positive definite** if the corresponding kernel $K$ is p.d.

**Theorem (Herglotz)**

A sequence $\{a_n\}_{n \in \mathbb{Z}}$ is p.d. if and only if it is the Fourier-Stieltjes transform of a positive measure $\mu \in M(\mathbb{T})$, the set of finite Borel measures on the torus $[0, 2\pi]$ with 0 and $2\pi$ identified.
Fourier-Stieltjes transform on the torus

- Let $\mathbb{T}$ the torus $[0, 2\pi]$ with 0 and $2\pi$ identified
- $C(\mathbb{T})$ the set of continuous functions on $\mathbb{T}$
- $M(\mathbb{T})$ the finite complex Borel measures\(^2\) on $\mathbb{T}$
- $M(\mathbb{T})$ can be identified as the dual space $(C(\mathbb{T}))^*$: for any continuous/bounded linear functional $\psi : C(\mathbb{T}) \to \mathbb{C}$ there exists $\mu \in M(\mathbb{T})$ such that $\psi(f) = \frac{1}{2\pi} \int_{\mathbb{T}} f(t) d\mu(t)$ (Riesz theorem).

**Definition (Fourier-Stieltjes coefficients)**

For any $\mu \in M(\mathbb{T})$, the Fourier-Stieltjes coefficients of $\mu$ is the sequence:

$$\forall n \in \mathbb{Z}, \quad \hat{\mu}(n) = \frac{1}{2\pi} \int_{\mathbb{T}} e^{-int} d\mu(t)$$

This extends the standard Fourier transform for integrable functions by taking $d\mu(t) = f(t)dt$.

\(^2\) a measure defined on all open sets
Examples

- **Diagonal kernel:**
  \[
  \mu = dt, \quad a_n = \hat{\mu}(n) = \frac{1}{2\pi} \int_T e^{int} dt = \begin{cases} 
  1 & \text{if } n = 0, \\
  0 & \text{otherwise.}
\end{cases}
  \]

  The resulting kernel is \( K(x, t) = \mathbf{1}(x = t) \).

- **Constant kernel:** for \( C \geq 0 \),
  \[
  \mu = 2\pi C \delta_0, \quad a_n = \hat{\mu}(n) = C \int_T e^{int} \delta_0(t) = C,
  \]

  resulting in \( K(x, t) = C \)
Proof of Herglotz’s theorem: ⇐

If \( a_n = \hat{\mu}(n) \) for \( \mu \in M(\mathbb{T}) \) positive, then for any \( n \in \mathbb{N} \), \( x_1, \ldots, x_n \in \mathbb{Z} \) and \( z_1, \ldots, z_n \in \mathbb{R} \) (or \( \mathbb{C} \)):

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} z_i \bar{z}_j a_{x_i-x_j} = \frac{1}{2\pi} \sum_{i=1}^{n} \sum_{j=1}^{n} z_i \bar{z}_j \int_{\mathbb{T}} e^{-i(x_i-x_j)t} d\mu(t)
\]

\[
= \frac{1}{2\pi} \sum_{i=1}^{n} \sum_{j=1}^{n} z_i \bar{z}_j \int_{\mathbb{T}} e^{-ix_it} e^{ix_jt} d\mu(t)
\]

\[
= \frac{1}{2\pi} \int_{\mathbb{T}} \left| \sum_{j=1}^{n} z_j e^{-ix_jt} \right|^2 d\mu(t)
\]

\( \geq 0. \)
Proof of Herglotz’s theorem: ⇒ (1/4)

- Let \( \{a_n\}_{n \in \mathbb{Z}} \) a p.d. sequence
- For a given \( t \in \mathbb{R} \) and \( N \in \mathbb{N} \) let \( \{z_n\}_{n \in \mathbb{Z}} \) be
  \[
  z_n = \begin{cases} 
  e^{int} & \text{if } |n| \leq N, \\ 
  0 & \text{otherwise.}
  \end{cases}
  \]
- Since \( \{a_n\}_{n \in \mathbb{Z}} \) is p.d. we get:
  \[
  0 \leq \sum_{k=-N}^{N} \sum_{l=-N}^{N} a_{k-l} z_k \bar{z}_l = \sum_{k=-N}^{N} \sum_{l=-N}^{N} a_{k-l} e^{i(k-l)t} \\
  = \sum_{k=-2N}^{2N} (2N + 1 - |k|) a_k e^{ikt} \\
  = (2N + 1) \sum_{k \in \mathbb{Z}} \max \left(0, 1 - \frac{|k|}{2N + 1}\right) a_k e^{ikt}
  \]

\[\sigma_{2N}(t)\]
Proof of Herglotz’s theorem: ⇒ (2/4)

- \( d\mu_N = \sigma_N(t)dt \) is a positive measure (for \( N \) even) and satisfies

\[
\hat{\mu}_N(n) = \frac{1}{2\pi} \sum_{j=-N}^{N} a_j \left( 1 - \frac{|j|}{N+1} \right) \int_{\mathbb{T}} e^{i(n-j)t} = a_n \max \left( 0, 1 - \frac{|n|}{N+1} \right)
\]

- Moreover

\[
\| \mu_N \|_{M(\mathbb{T})} = \sup_{\| f \|_{\infty} \leq 1} \int_{\mathbb{T}} f(t)\sigma_N(t)dt
\]

\[
= \int_{\mathbb{T}} \sigma_N(t)dt \quad \text{(take } f = 1 \text{ because } \sigma_N(t) \geq 0)\]

\[
= \sum_{n=-N}^{N} \int_{\mathbb{T}} a_n \left( 1 - \frac{|n|}{N+1} \right) e^{int} dt
\]

\[
= a_0
\]
Proof of Herglotz’s theorem: ⇒ (3/4)

For any trigonometric polynomial of the form
\[ P(t) = \sum_{k=-K}^{K} b_k e^{ikt}, \]
with Fourier coefficient \( \hat{P}(n) = b_n \), we have

\[
\lim_{N \to +\infty} \int_{\mathbb{T}} P(t) d\mu_N(t) = \lim_{N \to +\infty} \sum_{k=-K}^{K} \sum_{n=-N}^{N} \int_{\mathbb{T}} a_n b_k \left( 1 - \frac{|n|}{N + 1} \right) e^{i(n-k)t} dt
\]

\[
= \sum_{k=-K}^{K} a_k b_k \lim_{N \to +\infty} \left( 1 - \frac{|n|}{N + 1} \right)
\]

\[
= \sum_{k=-K}^{K} a_k b_k
\]

\[
= \sum_{k \in \mathbb{Z}} a_k \hat{P}(k)
\]
Proof of Herglotz’s theorem: \( \Rightarrow (4/4) \)

- This shows that \( \Psi(P) = \sum_{k \in \mathbb{Z}} a_k \hat{P}(k) \) is a linear functional over trigonometric polynomials, with norm \( \leq a_0 \)
- It can be extended to all continuous functions because trigonometric polynomials are dense in \( C(\mathbb{T}) \)
- By Riesz representation theorem, there exists a measure \( \mu \in M(\mathbb{T}) \) such that \( \| \mu \|_{M(\mathbb{T})} \leq a_0 \)

\[ \forall f \in C(\mathbb{T}), \quad \Psi(f) = \int_{\mathbb{T}} f(t) d\mu(t) \]

- Taking \( f(t) = e^{int} \) gives

\[ \hat{\mu}(n) = \int_{\mathbb{T}} e^{int} d\mu(t) = \Psi(e^{int}) = a_n \]

- Furthermore \( \mu \) is a positive measure because if \( f \geq 0 \)

\[ \int_{\mathbb{T}} f(t) d\mu(t) = \Psi(f) = \lim_{n \to +\infty} \Psi(P_n) = \lim_{n, k \to +\infty} \Psi_k(P_n) \geq 0 \]
Translation invariant kernels on $\mathbb{R}^d$

**Definition**

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

$$\forall x, y \in \mathbb{R}^d, \quad K(x, y) = \varphi(x - y)$$

for some function $\varphi : \mathbb{R}^d \mapsto \mathbb{R}$. Such a function $\varphi$ is called positive definite if the corresponding kernel $K$ is p.d.
Translation invariant kernels on $\mathbb{R}^d$

**Definition**

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

$$\forall x, y \in \mathbb{R}^d, \quad K(x, y) = \varphi(x - y)$$

for some function $\varphi : \mathbb{R}^d \to \mathbb{R}$. Such a function $\varphi$ is called positive definite if the corresponding kernel $K$ is p.d.

**Theorem (Bochner)**

A continuous function $\varphi : \mathbb{R}^d \to \mathbb{R}$ is p.d. if and only if it is the Fourier-Stieltjes transform of a symmetric and positive finite Borel measure $\mu \in M(\mathbb{R}^d)$. 
Fourier-Stieltjes transform on $\mathbb{R}^d$

- $C_0(\mathbb{R}^d)$ the set of continuous functions on $\mathbb{R}^d$ that vanish at infinity
- $M(\mathbb{R}^d)$ the finite complex Borel measures on $\mathbb{R}^d$
- $M(\mathbb{R}^d)$ can be identified as the dual space $(C_0(\mathbb{R}^d))^*$: for any continuous/bounded linear functional $\psi : C_0(\mathbb{R}^d) \to \mathbb{C}$ there exists $\mu \in M(\mathbb{R}^d)$ such that $\psi(f) = \int_{\mathbb{R}^d} f(t) d\mu(t)$ (Riesz theorem).

**Definition (Fourier-Stieltjes transform)**

For any $\mu \in M(\mathbb{R}^d)$, the Fourier-Stieltjes transform of $\mu$ is the function:

$$\forall \omega \in \mathbb{R}^d, \quad \hat{\mu}(\omega) = \int_{\mathbb{R}^d} e^{-i\omega^\top x} d\mu(x)$$
Fourier-Stieltjes transform on \( \mathbb{R}^d \)

- This extends the standard Fourier transform for integrable functions by taking \( d\mu(x) = f(x)dx \).
- For \( \mu \in M(\mathbb{R}^d) \), \( \hat{\mu} \) is still uniformly continuous, but \( \hat{\mu}(\omega) \) does not necessarily go to 0 at infinity (e.g., take the Dirac \( \mu = \delta_0 \), then \( \hat{\mu}(\omega) = 1 \) for all \( \omega \)).
- Parseval’s formula: if \( \mu \in M(\mathbb{R}^d) \), and both \( g, \hat{g} \) are in \( L^1(\mathbb{R}^d) \), then
  \[
  \int_{\mathbb{R}^d} g(x) d\mu(x) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{g}(\omega)\hat{\mu}(-\omega) d\omega.
  \]
  In particular, if \( g \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d) \),
  \[
  \int_{\mathbb{R}^d} g(x)^2 dx = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{g}(\omega)^2 d\omega.
  \]
Proof of Bochner’s theorem: \(\iff\)

If \(\varphi = \hat{\mu}\) for some \(\mu \in M(\mathbb{T})\) positive, then for any \(n \in \mathbb{N}\), \(x_1, \ldots, x_n \in \mathbb{R}^d\) and \(z_1, \ldots, z_n \in \mathbb{R}\) (or \(\mathbb{C}\)) :

\[
\begin{align*}
\sum_{i=1}^n \sum_{j=1}^n z_i \bar{z}_j \varphi (x_i - x_j) &= \sum_{i=1}^n \sum_{j=1}^n z_i \bar{z}_j \int_{\mathbb{R}^d} e^{-i(x_i - x_j)^\top t} d\mu(t) \\
&= \sum_{i=1}^n \sum_{j=1}^n z_i \bar{z}_j \int_{\mathbb{R}^d} e^{-ix_i^\top t} e^{ix_j^\top t} d\mu(t) \\
&= \int_{\mathbb{R}^d} |\sum_{j=1}^n z_j e^{-ix_j^\top t}|^2 d\mu(t) \\
&\geq 0.
\end{align*}
\]

If \(\mu\) is symmetric then, in addition, \(\varphi\) is real-valued. \(\square\)
Lemma

Let \( \varphi : \mathbb{R} \to \mathbb{R} \) continuous. If there exists \( C \geq 0 \) such that

\[
\left| \frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(\xi) \varphi(-\xi) d\xi \right| \leq C \sup_{x \in \mathbb{R}} |g(x)|
\]

for every continuous function \( g \in L^1(\mathbb{R}) \) such that \( \hat{g} \) is continuous and has compact support, then \( \varphi \) is the Fourier-Stieljes transform of a measure \( \mu \in M(\mathbb{R}) \).

Proof: Let \( \mathcal{G} \subset C_0(\mathbb{R}) \) be the set of functions \( g \in L^1(\mathbb{R}) \) such that \( \hat{g} \) is continuous and has compact support. \( \Psi : g \mapsto \frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(\xi) \varphi(-\xi) d\xi \) is linear and continuous on \( \mathcal{G} \), and can be extended to \( C_0(\mathbb{R}) \) by density of \( \mathcal{G} \). By Riesz theorem, there exists \( \mu \in M(\mathbb{R}) \) such that

\[
\Psi(g) = \int_{\mathbb{R}} g(x) d\mu(x) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(\xi) \hat{\mu}(-\xi) d\xi,
\]

using Parceval’s formula for the second equality. This must hold for all \( g \), so \( \varphi = \hat{\mu} \). □

Note: the converse is also true.
Proof of Bochner’s theorem: ⇒ (2/5)

- We consider $d = 1$. Generalization to $d > 1$ is trivial.
- Let $\varphi : \mathbb{R} \to \mathbb{R}$ continuous and p.d.
- For any $\lambda > 0$, the sequence $\{\varphi(n\lambda)\}_{n \in \mathbb{Z}}$ is p.d., so by Herglotz’s theorem there exists a positive measure $\mu_\lambda \in M(\mathbb{T})$ such that
  \[
  \varphi(\lambda n) = \hat{\mu}_\lambda(n),
  \]
  and $\|\mu_\lambda\|_{M(\mathbb{T})} = \hat{\mu}_\lambda(0) = \varphi(0)$.
- Let $g \in L^1(\mathbb{R})$ continuous such that $\hat{g}$ is continuous and has compact support.
- For any $\epsilon > 0$ there exists $\lambda > 0$ such that
  \[
  \left| \frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(\xi) \varphi(-\xi) d\xi \right| < \left| \frac{\lambda}{2\pi} \sum_{n \in \mathbb{Z}} \hat{g}(\lambda n) \varphi(-\lambda n) \right| + \epsilon,
  \]
  by approximating the integral by its Riemann sums (where the width of each rectangle is $\lambda$).
Proof of Bochner’s theorem: \( \Rightarrow (3/5) \)

- For \( t \in \mathbb{T} \) let:
  \[
  G_\lambda(t) = \sum_{m \in \mathbb{Z}} g \left( \frac{t + 2\pi m}{\lambda} \right)
  \]

- Given the regularity and decay of \( g \), we can find a sufficiently small \( \lambda \) to ensure
  \[
  \sup_{t \in \mathbb{T}} |G_\lambda(t)| \leq \sup_{x \in \mathbb{R}} |g(x)| + \epsilon
  \]
Proof of Bochner’s theorem: $\Rightarrow (3/5)$

- In addition, for any $n \in \mathbb{Z}$:

$$\hat{G}_\lambda(n) = \frac{1}{2\pi} \int_{\mathbb{T}} e^{-int} G_\lambda(t) dt$$

$$= \frac{1}{2\pi} \sum_{m \in \mathbb{Z}} \int_{0}^{2\pi} e^{-int} g \left( t + \frac{2\pi m}{\lambda} \right) dt$$

$$= \frac{\lambda}{2\pi} \sum_{m \in \mathbb{Z}} \int_{\frac{2\pi m}{\lambda}}^{\frac{2\pi(m+1)}{\lambda}} e^{-in(\lambda u + 2\pi m)} g(u) du$$

$$= \frac{\lambda}{2\pi} \sum_{m \in \mathbb{Z}} \int_{\frac{2\pi m}{\lambda}}^{\frac{2\pi(m+1)}{\lambda}} e^{-in\lambda u} g(u) du$$

$$= \frac{\lambda}{2\pi} \int_{\mathbb{R}} e^{-in\lambda u} g(u) du$$

$$= \frac{\lambda}{2\pi} \hat{g}(\lambda n)$$
Proof of Bochner’s theorem: \( \Rightarrow (4/5) \)

This gives:

\[
\left| \frac{\lambda}{2\pi} \sum_{n \in \mathbb{Z}} \hat{g}(\lambda n) \varphi(-\lambda n) \right| = \left| \sum_{n \in \mathbb{Z}} \hat{G}_\lambda(n) \hat{\mu}_\lambda(-n) \right|
\]

\[
= \left| \frac{1}{2\pi} \int_\mathbb{T} G_\lambda(t) d\mu_\lambda(t) \right| \quad \text{(Parceval)}
\]

\[
\leq \| \mu_\lambda \|_{M(\mathbb{T})} \sup_{t \in \mathbb{T}} |G_\lambda(t)|
\]

\[
\leq C \sup_{t \in \mathbb{T}} |G_\lambda(t)|
\]

\[
\leq C \sup_{x \in \mathbb{R}} |g(x)| + C \epsilon
\]

with \( C = \varphi(0) \).
Proof of Bochner’s theorem: \( \Rightarrow (5/5) \)

- Putting it all together gives:

\[
\left| \frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(\xi)\varphi(-\xi) d\xi \right| < C \sup_{x \in \mathbb{R}} |g(x)| + (C + 1) \epsilon
\]

- This is true for all \( \epsilon > 0 \) which implies

\[
\left| \frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(\xi)\varphi(-\xi) d\xi \right| \leq C \sup_{x \in \mathbb{R}} |g(x)|
\]

- We conclude from the lemma that \( \varphi = \hat{\mu} \) for some \( \mu \in M(\mathbb{R}) \), which satisfies

\[
\frac{1}{2\pi} \int_{\mathbb{R}} \hat{g}(\xi)\varphi(-\xi) d\xi = \int_{\mathbb{R}} g(x) d\mu(x)
\]

- When \( g \geq 0 \), this is approximated by \( \frac{1}{2\pi} \int_{\mathbb{T}} G_\lambda(t)d\mu_\lambda(t) \) for small \( \lambda \), which is \( \geq 0 \) because \( \mu_\lambda \) is a positive measure and \( G_\lambda \geq 0 \) like \( g \). Consequently, \( \mu \) is a positive measure. \( \square \)
Theorem

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

$$\| f \|^2_{\mathcal{H}} := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \left| \hat{f}(\omega) \right|^2 \frac{\hat{\varphi}(\omega)}{\hat{\varphi}(\omega)} d\omega < +\infty,$$

endowed with the inner product:

$$\langle f, g \rangle_{\mathcal{H}} := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{f}(\omega) \overline{\hat{g}(\omega)} \frac{\hat{\varphi}(\omega)}{\hat{\varphi}(\omega)} d\omega$$

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

- $\mathcal{H}$ is a Hilbert space: exercise.

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

$$\hat{K}_x(\omega) = \int e^{-i\omega^\top u} \varphi(u - x) du = e^{-i\omega^\top x} \hat{\varphi}(\omega).$$

- This leads to $K_x \in \mathcal{H}$, because:

$$\int_{\mathbb{R}^d} \left| \hat{K}_x(\omega) \right|^2 \frac{1}{\hat{\varphi}(\omega)} \leq \frac{1}{\int_{\mathbb{R}^d} \left| \hat{\varphi}(\omega) \right| < \infty},$$

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

$$\langle f, K_x \rangle_{\mathcal{H}} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \frac{\hat{K}_x(\omega) \hat{f}(\omega)}{\hat{\varphi}(\omega)} d\omega = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{f}(\omega) e^{-i\omega^\top x} d\omega = f(x)$$

$\square$
Example

Gaussian kernel

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

corresponds to:

$$\varphi(t) = e^{-\frac{t^2}{2\sigma^2}}$$

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

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 \textit{infinitely differentiable} with all derivatives in $L^2$. 
Example

Laplace kernel

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

corresponds to:

\[ \varphi(t) = \frac{1}{2} e^{-\gamma |t|} \]

\[ \hat{\varphi}(\omega) = \frac{\gamma}{\gamma^2 + \omega^2} \]

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:

\[ \varphi(t) = \frac{\sin(\Omega t)}{\pi t} \]

\[ \hat{\varphi}(\omega) = 1_{[-\Omega, \Omega]}(\omega) \]

and

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

the set of functions whose spectrum is included in \([-\Omega, \Omega]\).
Recap on Green, Mercer, Bochner families

Up to specific assumptions for each of the following kernel families,

<table>
<thead>
<tr>
<th>Kernel</th>
<th>RKHS ( \mathcal{H} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Green func. of ( D^*D )</td>
</tr>
<tr>
<td>Mercer</td>
<td>( \sum_{k=1}^{\infty} \lambda_k \psi_k(x)\psi_k(y) )</td>
</tr>
<tr>
<td>Fourier</td>
<td>( \kappa(x-y) )</td>
</tr>
</tbody>
</table>

\( \mathcal{H} = L_2(\mathcal{X}) \) with \( \langle Df, Dg \rangle_{L_2(\mathcal{X})} \)

\| \begin{align*} f &= \sum_{k=1}^{\infty} a_k \psi_k : \sum_{k=1}^{\infty} \frac{a_k^2}{\lambda_k} < +\infty \\ f &\in L_2(\mathbb{R}^d) : \int \frac{|\hat{f}(\omega)|^2}{\hat{\kappa}(\omega)} d\omega < +\infty \end{align*} \|

+ continuous + integrable
Recap on Green, Mercer, Bochner families

Up to specific assumptions for each of the following kernel families,

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Squared Norm $| \cdot |_H^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>$| Df |_{L_2(\mathcal{X})}^2$</td>
</tr>
<tr>
<td>Mercer</td>
<td>$\sum_{k=1}^{\infty} \frac{a_k^2}{\lambda_k}$ for $f = \sum_{k=1}^{\infty} a_k \psi_k$</td>
</tr>
<tr>
<td>Fourier</td>
<td>$\frac{1}{(2\pi)^d} \int \frac{</td>
</tr>
</tbody>
</table>
The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
  - Green kernels
  - Mercer kernels
  - Convergence rates of KRR for Mercer kernels
  - Shift-invariant kernels
  - Generalization to semigroups
- Kernels for probabilistic 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 \(\varphi : S \to \mathbb{R}\) 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 \(\varphi : \mathbb{R}^d \to \mathbb{R}\) is p.d. if the function

\[
K(x, y) = \varphi(x - y)
\]

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 $(S, +, \ast)$ is called a semicharacter if

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

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

Remarks

- If $\ast$ is the identity, a semicharacter is automatically real-valued.
- If $(S, +)$ is an abelian group and $s^\ast = -s$, 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:

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

**Proof**

Direct from definition, e.g.,

\[
\sum_{i,j=1}^{n} a_i \overline{a_j} \rho(x_i + x_j^*) = \sum_{i,j=1}^{n} a_i \overline{a_j} \rho(x_i) \overline{\rho(x_j)} \geq 0.
\]

**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

Definition

- An function \( \alpha : S \to \mathbb{R} \) 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 \( f : S \to \mathbb{R} \) 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 \((S, +, \ast)\) an abelian semigroup with involution. A function \( \varphi : S \to \mathbb{R} \) 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: \((R_+, +, \text{ld})\)

**Semicharacters**

- \(S = (R_+, +, \text{ld})\) 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:
  \[
  s \in R_+ \mapsto \rho_a(s) = e^{-as},
  \]
  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 \(h(x + y) = h(x)h(y)\).
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 \( \mathcal{M}_+^b (\mathbb{R}) \).
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).
We can represent any bag-of-point $x$ as a finite measure on $U$:

$$x = \sum_{i} a_i \delta_{x_i},$$

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 $(\mathcal{M}_+^b(U), +, \text{Id})$ of bounded Radon measures on $U$ and to search for p.d. functions $\varphi$ on this semigroup.
Semicharacters

For any Borel measurable function $f : \mathcal{U} \to \mathbb{R}$ the function

$$\rho_f : \mathcal{M}^b_+ (\mathcal{U}) \to \mathbb{R}$$

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 $f : \mathcal{U} \to \mathbb{R}$ such that $\rho = \rho_f$.

No such characterization for non-continuous characters, even bounded.
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:

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

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

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

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

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

Then the following function is a p.d. kernel on \( \mathcal{M}_+^V (U) \), 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

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.

\begin{align*}
\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
\end{align*}
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
## 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.
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. Kernel tricks
3. Kernel Methods: Supervised Learning
5. The Kernel Jungle
   - Green, Mercer, Herglotz, Bochner and friends
   - Kernels for probabilistic models
   - Kernels for biological sequences
   - Kernels for graphs
   - Kernels on graphs
6. Open Problems and Research Topics
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 representation of image and sequence data representation.

Parametric model

A model is a family of distributions

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

The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic models
  - Fisher kernel
  - Mutual information kernels
  - Marginalized kernels
- Kernels for biological sequences
- Kernels for graphs
- Kernels on graphs
Fisher kernel

Definition

- Fix a parameter $\theta_0 \in \Theta$ (obtained for instance by maximum likelihood over a training set).
- For each sequence $x$, compute the Fisher score vector:
  \[
  \Phi_{\theta_0}(x) = \nabla_\theta \log P_\theta(x)_{\theta=\theta_0},
  \]
  which can be interpreted as the local contribution of each parameter.
- Form the kernel (Jaakkola et al., 2000):
  \[
  K(x, x') = \Phi_{\theta_0}(x)^\top I(\theta_0)^{-1}\Phi_{\theta_0}(x'),
  \]
  where $I(\theta_0) = \mathbb{E} \left[ \Phi_{\theta_0}(x)\Phi_{\theta_0}(x)^\top \right]$ is the Fisher information matrix.

Note: when $\theta_0$ is the ML estimator, $\mathbb{E}[\Phi_{\theta_0}(x)] = 0$ and $I(\theta_0)$ is a covariance matrix.
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 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).
The Fisher kernel is invariant under change of parametrization.

Consider indeed a 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

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

We conclude by noticing that \( \mathbf{I}(\lambda_0)^{-1} = \mathbf{J}^{-1} \mathbf{I}(\theta_0)^{-1} \mathbf{J}^\top^{-1} \):

\[
\mathcal{K}(\mathbf{x}, \mathbf{x}') = \Phi_{\theta_0}(\mathbf{x})^\top \mathbf{I}(\theta_0)^{-1} \Phi_{\theta_0}(\mathbf{x}') = \Phi_{\lambda_0}(\mathbf{x})^\top \mathbf{I}(\lambda_0)^{-1} \Phi_{\lambda_0}(\mathbf{x}').
\]
Fisher kernel in practice

- $\Phi_{\theta_0}(x)$ can be computed explicitly for many models (e.g., HMMs), where the model is first estimated from data.
- $I(\theta_0)$ 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 $\varphi_{\theta_0}(x) = I(\theta_0)^{-1/2}\Phi_{\theta_0}(x)$. They are explicitly computed and correspond to an explicit embedding: $K(x, x') = \varphi_{\theta_0}(x)^\top \varphi_{\theta_0}(x')$. 
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

$$\frac{\partial \log P_{\theta}(x)}{\partial \mu} = \alpha (x - \mu), \quad \frac{\partial \log P_{\theta}(x)}{\partial \alpha} = \frac{1}{2} \left[ \frac{1}{\alpha} - (x - \mu)^2 \right],$$

and (exercise)

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

The Fisher vector is then

$$\varphi_{\theta}(x) = \begin{pmatrix} (x - \mu)/\sigma \\ (1/\sqrt{2})(1 - (x - \mu)^2/\sigma^2) \end{pmatrix}.$$
Now consider an i.i.d. data model over a set of data points $x_1, \ldots, x_n$ all distributed according to $\mathcal{N}(\mu, \sigma^2)$:

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

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.

$$\varphi(x_1, \ldots, x_n) = \sum_{i=1}^{n} \varphi(x_i) = n \left( \frac{(\hat{\mu} - \mu)/\sigma}{(\sigma^2 - \hat{\sigma}^2)/(\sqrt{2}\sigma^2)} \right),$$

- where

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad \text{and} \quad \hat{\sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2.$$
**Patch extraction and description stage:**
In various contexts, images may be described as a set of patches $x_1, \ldots, x_n$ 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 $\varphi(x_i)$, typically in a high-dimensional space.

**Pooling stage:** For example, sum pooling

$$\varphi(x_1, \ldots, x_n) = \sum_{i=1}^{n} \varphi(x_i).$$

Fisher vectors with a Gaussian Mixture Model (GMM) is a simple and effective aggregation technique (Perronnin and Dance, 2007).
Let $\theta = (\pi_j, \mu_j, \Sigma_j)_{j=1 \ldots, k}$ be the parameters of a GMM with $k$ Gaussian components. Then, the probabilistic model is given by

$$P_\theta(x) = \sum_{j=1}^{k} \pi_j \mathcal{N}(x; \mu_j, \Sigma_j).$$

### Remarks

- Each mixture component corresponds to a visual word, with a mean, variance, and mixing weight.
- Diagonal covariances $\Sigma_j = \text{diag} (\sigma_{j1}, \ldots, \sigma_{jp}) = \text{diag} (\sigma_j)$ 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 (3/5)

After cumbersome calculations (exercise), we obtain \( \varphi_\theta(x_1, \ldots, x_n) = \left[ \varphi_{\pi_1}(X), \ldots, \varphi_{\pi_p}(X), \varphi_{\mu_1}(X)^T, \ldots, \varphi_{\mu_p}(X)^T, \varphi_{\sigma_1}(X)^T, \ldots, \varphi_{\sigma_p}(X)^T \right]^T \),

with

\[
\varphi_{\mu_j}(X) = \frac{1}{n \sqrt{\pi_j}} \sum_{i=1}^{n} \gamma_{ij} (x_i - \mu_j) / \sigma_j \\
\varphi_{\sigma_j}(X) = \frac{1}{n \sqrt{2\pi_j}} \sum_{i=1}^{n} \gamma_{ij} \left[ (x_i - \mu_j)^2 / \sigma_j^2 - 1 \right],
\]

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 \( j \):

\[
\gamma_{ij} = \frac{\pi_j \mathcal{N}(x_i; \mu_j, \sigma_j)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(x_i; \mu_l, \sigma_l)}.
\]
Finally, we also have the following interpretation of encoding first and second-order statistics:

\[ \varphi_{\mu_j}(X) = \frac{\gamma_j}{\sqrt{\pi_j}} \frac{(\hat{\mu}_j - \mu_j)}{\sigma_j} \]

\[ \varphi_{\sigma_j}(X) = \frac{\gamma_j}{\sqrt{2\pi_j}} \frac{(\hat{\sigma}_j^2 - \sigma_j^2)}{\sigma_j^2}, \]

with

\[ \gamma_j = \sum_{i=1}^{n} \gamma_{ij} \quad \text{and} \quad \hat{\mu}_j = \frac{1}{\gamma_j} \sum_{i=1}^{n} \gamma_{ij} x_i \quad \text{and} \quad \hat{\sigma}_j = \frac{1}{\gamma_j} \sum_{i=1}^{n} \gamma_{ij} (x_i - \mu_j)^2. \]

The component \( \varphi_{\pi}(X) \) 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 \( x_i \)'s.
FVs were state-of-the-art image representations before the revival of convolutional neural networks in 2012.
FVs were state-of-the-art image representations before the revival of convolutional neural networks in 2012. This is an unsupervised image representation of high dimension. They remain competitive among unsupervised methods, see the following table from Bojanowski and Joulin, 2017.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random (Noroozi &amp; Favaro, 2016)</td>
<td>12.0</td>
</tr>
<tr>
<td>SIFT+FV (Sánchez et al., 2013)</td>
<td>55.6</td>
</tr>
<tr>
<td>Wang &amp; Gupta (2015)</td>
<td>29.8</td>
</tr>
<tr>
<td>Doersch et al. (2015)</td>
<td>30.4</td>
</tr>
<tr>
<td>Zhang et al. (2016)</td>
<td>35.2</td>
</tr>
<tr>
<td>¹Noroozi &amp; Favaro (2016)</td>
<td>38.1</td>
</tr>
<tr>
<td>BiGAN (Donahue et al., 2016)</td>
<td>32.2</td>
</tr>
<tr>
<td>NAT</td>
<td>36.0</td>
</tr>
</tbody>
</table>

Table 3. Comparison of the proposed approach to state-of-the-art unsupervised feature learning on ImageNet. A full multi-layer perceptron is retrained on top of the features. We compare to several self-supervised approaches and an unsupervised approach, i.e., BiGAN (Donahue et al., 2016). ¹Noroozi & Favaro (2016)
Relation to classification with generative models (1/3)

Assume that we have a generative probabilistic model $P_\theta$ to model random variables $(X, Y)$ where $Y$ is a label in $\{1, \ldots, p\}$.

Assume that the marginals $P_\theta(Y = k) = \pi_k$ are among the model parameters $\theta$, which we can also parametrize as

$$P_\theta(Y = k) = \pi_k = \frac{e^{\alpha_k}}{\sum_{k' = 1}^{p} e^{\alpha_{k'}}}.$$

The classification of a new point $x$ can be obtained via Bayes’ rule:

$$\hat{y}(x) = \arg\max_{k = 1, \ldots, p} P_\theta(Y = k|x),$$

where $P_\theta(Y = k|x)$ is short for $P_\theta(Y = k|X = x)$ and

$$P_\theta(Y = k|x) = P_\theta(x|Y = k)P_\theta(Y = k)/P_\theta(x)$$

$$= P_\theta(x|Y = k)\pi_k / \sum_{k' = 1}^{p} P_\theta(x|Y = k')\pi_{k'}.$$
Relation to classification with generative models (2/3)

Then, consider the Fisher score

\[
\nabla_\theta \log P_\theta(x) = \frac{1}{P_\theta(x)} \nabla_\theta P_\theta(x)
\]

\[
= \frac{1}{P_\theta(x)} \nabla_\theta \sum_{k=1}^{p} P_\theta(x, Y = k)
\]

\[
= \frac{1}{P_\theta(x)} \sum_{k=1}^{p} P_\theta(x, Y = k) \nabla_\theta \log P_\theta(x, Y = k)
\]

\[
= \sum_{k=1}^{p} P_\theta(Y = k|x)[\nabla_\theta \log \pi_k + \nabla_\theta \log P_\theta(x|Y = k)].
\]

In particular (exercise)

\[
\frac{\partial \log P_\theta(x)}{\partial \alpha_k} = P_\theta(Y = k|x) - \pi_k.
\]
The first $p$ elements in the Fisher score are given by class posteriors minus a constant

$$
\varphi_\theta(x) = [P_\theta(Y = 1|x) - \pi_1, \ldots, P_\theta(Y = p|x) - \pi_p, \ldots].
$$

Consider a multi-class linear classifier on $\varphi_\theta(x)$ such that for class $k$

- The weights are zero except one for the $k$-th position;
- The intercept $b_k$ be $\pi_k$;

Then,

$$
\hat{y}(x) = \arg\max_{k=1,\ldots,p} \varphi_\theta(x)^\top w_k + b_k
$$

$$
\hat{y}(x) = \arg\max_{k=1,\ldots,p} P_\theta(Y = k|x).
$$

Bayes’ rule is implemented via this simple classifier using Fisher kernel.
The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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):

  $$K(x, x') = \int_{\theta \in \Theta} P_{\theta}(x)P_{\theta}(x')w(d\theta).$$

- No explicit computation of a finite-dimensional feature vector.
- $K(x, x') = \langle \varphi(x), \varphi(x') \rangle_{L^2(w)}$ with

  $$\varphi(x) = (P_{\theta}(x))_{\theta \in \Theta}.$$
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 $x = 001$ and $x' = 1010$ is:

\[
\begin{align*}
\begin{cases}
P_\theta(x) & = \theta (1 - \theta)^2, \\
P_\theta(x') & = \theta^2 (1 - \theta)^2,
\end{cases}
\end{align*}
\]

\[
K(x, x') = \int_0^1 \theta^3 (1 - \theta)^4 d\theta = \frac{3!4!}{8!} = \frac{1}{280}.
\]
The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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 $x \in \mathcal{X}$, let a latent variable $y \in \mathcal{Y}$ be associated probabilistically through a conditional probability $P_x(dy)$.
- Let $K_Z$ be a kernel for the complete data $z = (x, y)$.
- Then, the following kernel is a valid kernel on $\mathcal{X}$, called a marginalized kernel (Tsuda et al., 2002):

$$K_{\mathcal{X}}(x, x') := E_{P_x(dy) \times P_{x'}(dy')} K_Z(z, z')$$

$$= \int \int K_Z((x, y), (x', y')) P_x(dy) P_{x'}(dy').$$
Marginalized kernels: proof of positive definiteness

- $K_Z$ is p.d. on $\mathcal{Z}$. Therefore, there exists a Hilbert space $\mathcal{H}$ and $\Phi_Z : \mathcal{Z} \to \mathcal{H}$ such that:

$$K_Z (z, z') = \langle \Phi_Z (z), \Phi_Z (z') \rangle_{\mathcal{H}}.$$ 

- Marginalizing therefore gives:

$$K_X (x, x') = \mathbb{E}_{P_x(dy) \times P_{x'}(dy')} K_Z (z, z')$$

$$= \mathbb{E}_{P_x(dy) \times P_{x'}(dy')} \langle \Phi_Z (z), \Phi_Z (z') \rangle_{\mathcal{H}}$$

$$= \langle \mathbb{E}_{P_x(dy)} \Phi_Z (z), \mathbb{E}_{P_{x'}(dy')} \Phi_Z (z') \rangle_{\mathcal{H}},$$

therefore $K_X$ is p.d. on $\mathcal{X}$. □
Marginalized kernels: proof of positive definiteness

- \( K_Z \) is p.d. on \( \mathcal{Z} \). Therefore, there exists a Hilbert space \( \mathcal{H} \) and \( \Phi_Z : \mathcal{Z} \rightarrow \mathcal{H} \) such that:

\[
K_Z (z, z') = \langle \Phi_Z(z), \Phi_Z(z') \rangle_{\mathcal{H}}.
\]

- Marginalizing therefore gives:

\[
K_X (x, x') = \mathbb{E}_{P_x(dy) \times P_{x'}(dy')} K_Z (z, z')
\]

\[
= \mathbb{E}_{P_x(dy) \times P_{x'}(dy')} \langle \Phi_Z(z), \Phi_Z(z') \rangle_{\mathcal{H}}
\]

\[
= \langle \mathbb{E}_{P_x(dy)} \Phi_Z(z), \mathbb{E}_{P_{x'}(dy')} \Phi_Z(z') \rangle_{\mathcal{H}},
\]

therefore \( K_X \) is p.d. on \( \mathcal{X} \). \( \square \)

Of course, we make the right assumptions such that each operation above is valid, and all quantities are well defined.
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
5. The Kernel Jungle
   - Green, Mercer, Herglotz, Bochner and friends
   - Kernels for probabilistic models
   - Kernels for biological sequences
   - Kernels for graphs
   - Kernels on graphs
6. Open Problems and Research Topics
Outline

5 The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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, Watson, Wilkins and Franklin)
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

microtubules

mitochondrion

centriole

tissue

vesicle

cytoplasm

plasma membrane

rough endoplasmic reticulum

ribosomes

smooth endoplasmic reticulum

nucleus

nuclear envelope

nuclear pore

nucleolus

Golgi complex

lysosome

flagellum
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).

James Watson, Francis Crick, and Maurice Wilkins received the Nobel prize for this discovery in 1962. Key to this discovery were the X-ray crystallography images obtained by Rosalind Franklin.
Structure of DNA
The double helix
Central dogma

DNA → RNA → Proteins
Proteins
Genetic code

DNA = 4 letters (ATCG)

RNA = 4 letters (AUCG)

Protein = 20 letters (amino acids)

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: between 0.5 and 1 billion 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).
Cost of human genome sequencing

Cost per Genome

$100M $10M $1M $100K $10K $1K

Moore's Law

NIH National Human Genome Research Institute
genome.gov/sequencingcosts
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
N : Asparagin
H : Histidine
Y : Tyrosine
S : Serine
Q : Glutamine
L : Leucine
M : Methionine
R : Arginine
W : Tryptophane
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:
  - MASKATLLLAFTLLFATCIARHQQRQQQCNQCLQNIEA...
  - MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...
  - MALHTVILMLSSLPMLEAQNPEHANITIGEPITNETLGW...
  - ...

- Non-secreted proteins:
  - MAPPSVFAEVPQAQPVLVFKLADFDREDPDPRKVNLGVG...
  - MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...
  - MSISESYAKEIKTAFQRQFTDFPIEGEQFEDFLPIIGNP...
  - ...

Goal

- Build a classifier to predict whether new proteins are secreted or not.
Supervised classification with vector embedding

The idea

- Map each string $x \in \mathcal{X}$ to a vector $\Phi(x) \in \mathcal{F}$.
- Train a classifier for vectors on the images $\Phi(x_1), \ldots, \Phi(x_n)$ of the training set (nearest neighbor, linear perceptron, logistic regression, support vector machine...).
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 (gives 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
The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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 $x$ by a **fixed-length numerical vector** $\Phi(x) \in \mathbb{R}^n$. How to perform this embedding?
Vector embedding for strings

The idea
Represent each sequence $x$ by a fixed-length numerical vector $\Phi(x) \in \mathbb{R}^n$. 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)

$$r_j = \frac{1}{n-j} \sum_{i=1}^{n-j} h_i h_{i+j}$$
### The approach

Alternatively, index the feature space by fixed-length strings, i.e.,

$$\Phi(x) = (\Phi_u(x))_{u \in A^k}$$

where $\Phi_u(x)$ can be:

- the number of occurrences of $u$ in $x$ (without gaps): **spectrum kernel** (Leslie et al., 2002)
- the number of occurrences of $u$ in $x$ up to $m$ mismatches (without gaps): **mismatch kernel** (Leslie et al., 2004)
- the number of occurrences of $u$ in $x$ allowing gaps, with a weight decaying exponentially with the number of gaps: **substring kernel** (Lohdi et al., 2002)
Example: Spectrum kernel (1/4)

Kernel definition

- The 3-spectrum of

\[ x = \text{CGGSLIAMMWFGV} \]

is:

\[ (\text{CGG, GGS, GSL, SLI, LIA, IAM, AMM, MMW, MWF, WFG, FGV}) \].

- Let \( \Phi_u(x) \) denote the number of occurrences of \( u \) in \( x \). The \( k \)-spectrum kernel is:

\[
K(x, x') := \sum_{u \in A^k} \Phi_u(x) \Phi_u(x').
\]
Example: Spectrum kernel (2/4)

**Implementation**

- The computation of the kernel is formally a sum over $|\mathcal{A}|^k$ terms, but at most $|x| - k + 1$ terms are non-zero in $\Phi(x)$.

  Computation in $O(|x| + |x'|)$ with pre-indexation of the strings.

- Fast classification of a sequence $x$ in $O(|x|)$:

  $$f(x) = w \cdot \Phi(x) = \sum_{u} w_u \Phi_u(x) = |x| - k + 1 \sum_{i=1} w_{x_i...x_{i+k-1}}.$$  

**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.
If pre-indexation is not possible: retrieval tree (trie)

Consider the sequence ACGTTTAACGTAC.

The complexity for computing $K(x, x')$ becomes $O(k(|x| + |x'|))$.  

Example: Spectrum kernel (3/4)
Example: Spectrum kernel (4/4)

If pre-indexation is not possible: use a suffix tree

The complexity for computing $K(x, x')$ becomes $O(|x| + |x'|)$, but with a larger constant than with pre-indexation.
Definition

- For $1 \leq k \leq n \in \mathbb{N}$, we denote by $\mathcal{I}(k, n)$ the set of sequences of indices $i = (i_1, \ldots, i_k)$, with $1 \leq i_1 < i_2 < \ldots < i_k \leq n$.

- For a string $x = x_1 \ldots x_n \in \mathcal{X}$ of length $n$, for a sequence of indices $i \in \mathcal{I}(k, n)$, we define a substring as:

$$x(i) := x_{i_1} x_{i_2} \ldots x_{i_k}.$$ 

- The length of the substring is:

$$l(i) = i_k - i_1 + 1.$$
Example 2: Substring kernel (2/12)

Example

ABRACADABRA

- $i = (3, 4, 7, 8, 10)$
- $x(i) =$RADAR
- $l(i) = 10 - 3 + 1 = 8$
Example 2: Substring kernel (3/12)

The kernel

- Let $k \in \mathbb{N}$ and $\lambda \in \mathbb{R}^+$ fixed. For all $u \in \mathcal{A}^k$, let $\Phi_u : \mathcal{X} \to \mathbb{R}$ be defined by:

  $$\forall x \in \mathcal{X}, \quad \Phi_u(x) = \sum_{i \in \mathcal{I}(k, |x|): \ x(i) = u} \lambda^{l(i)}.$$

- The substring kernel is the p.d. kernel defined by:

  $$\forall (x, x') \in \mathcal{X}^2, \quad K_{k, \lambda}(x, x') = \sum_{u \in \mathcal{A}^k} \Phi_u(x) \Phi_u(x').$$
Example 2: Substring kernel (4/12)

Example

<table>
<thead>
<tr>
<th></th>
<th>ca</th>
<th>ct</th>
<th>at</th>
<th>ba</th>
<th>bt</th>
<th>cr</th>
<th>ar</th>
<th>br</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Phi_u(\text{cat})$</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\Phi_u(\text{car})$</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda^3$</td>
<td>$\lambda^2$</td>
<td>0</td>
</tr>
<tr>
<td>$\Phi_u(\text{bat})$</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\Phi_u(\text{bar})$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
K(\text{cat,cat}) &= K(\text{car,car}) = 2\lambda^4 + \lambda^6 \\
K(\text{cat,car}) &= \lambda^4 \\
K(\text{cat,bar}) &= 0
\end{align*}
\]
Example 2: Substring kernel (5/12)

Kernel computation

- We need to compute, for any pair $x, x' \in \mathcal{X}$, the kernel:

$$K_{k, \lambda}(x, x') = \sum_{u \in \mathcal{A}^k} \Phi_u(x) \Phi_u(x')$$

$$= \sum_{u \in \mathcal{A}^k} \sum_{i: x(i) = u} \sum_{i': x'(i') = u} \lambda^{l(i) + l(i')}.$$

- Enumerating the substrings is too slow (of order $|x|^k$).
Example 2: Substring kernel (6/12)

Kernel computation (cont.)

- For \( u \in A^k \) remember that:
  \[
  \Phi_u(x) = \sum_{i : x(i) = u} \lambda^{i_k - i_1 + 1}.
  \]

- Let now:
  \[
  \Psi_u(x) = \sum_{i : x(i) = u} \lambda^{|x| - i_1 + 1}.
  \]
Let us note $x_{[1,j]} = x_1 \ldots x_j$. A simple rewriting shows that, if we note $a \in \mathcal{A}$ the last letter of $u$ ($u = va$):

$$
\Phi_{va}(x) = \sum_{j \in [1,|x|]: x_j = a} \Psi_{v}(x_{[1,j-1]}) \lambda,
$$

and

$$
\Psi_{va}(x) = \sum_{j \in [1,|x|]: x_j = a} \Psi_{v}(x_{[1,j-1]}) \lambda^{|x|-j+1}.
$$
Moreover we observe that if the string is of the form $xa$ (i.e., the last letter is $a \in A$), then:

- If the last letter of $u$ is not $a$:
  \[
  \begin{align*}
  \Phi_u (xa) &= \Phi_u (x), \\
  \Psi_u (xa) &= \lambda \Psi_u (x).
  \end{align*}
  \]

- If the last letter of $u$ is $a$ (i.e., $u = va$ with $v \in A^{k-1}$):
  \[
  \begin{align*}
  \Phi_{va} (xa) &= \Phi_{va} (x) + \lambda \Psi_v (x), \\
  \Psi_{va} (xa) &= \lambda \Psi_{va} (x) + \lambda \Psi_v (x).
  \end{align*}
  \]
Let us now show how the function:

\[ B_k(x, x') := \sum_{u \in A^k} \Psi_u(x) \Psi_u(x') \]

and the kernel:

\[ K_k(x, x') := \sum_{u \in A^k} \Phi_u(x) \Phi_u(x') \]

can be computed recursively. We note that:

\[
\begin{align*}
B_0(x, x') &= K_0(x, x') = 1 & \text{for all } x, x' \\
B_k(x, x') &= K_k(x, x') = 0 & \text{if } \min(|x|, |x'|) < k
\end{align*}
\]
Example 2: Substring kernel (10/12)

Recursive computation of $B_k$

\[
B_k(xa, x') = \sum_{u \in A^k} \Psi_u(xa) \Psi_u(x') = \lambda \sum_{u \in A^k} \Psi_u(x) \Psi_u(x') + \lambda \sum_{v \in A^{k-1}} \Psi_v(x) \Psi_{va}(x')
\]

\[
= \lambda B_k(x, x') + \lambda \sum_{v \in A^{k-1}} \Psi_v(x) \left( \sum_{j \in [1, |x'|]} \Psi_v(x'_{[1,j-1]}) \lambda |x'|-j+1 \right)
\]

\[
= \lambda B_k(x, x') + \sum_{j \in [1, |x'|]} B_{k-1}(x, x'_{[1,j-1]}) \lambda |x'|-j+2
\]
Example 2: Substring kernel (11/12)

Recursive computation of $B_k$

$$B_k(xa, x'b)$$

$$= \lambda B_k(x, x'b) + \lambda \sum_{j \in [1, |x'|]: x'_j = a} B_{k-1}(x, x'_{[1,j-1]}) \lambda^{|x'| - j + 2}$$

$$+ \delta_{a=b} B_{k-1}(x, x') \lambda^2$$

$$= \lambda B_k(x, x'b) + \lambda (B_k(xa, x') - \lambda B_k(x, x')) + \delta_{a=b} B_{k-1}(x, x') \lambda^2$$

$$= \lambda B_k(x, x'b) + \lambda B_k(xa, x') - \lambda^2 B_k(x, x') + \delta_{a=b} B_{k-1}(x, x') \lambda^2.$$

The dynamic programming table can be filled in $O(k|x||x'|)$ operations.
Example 2: Substring kernel (12/12)

Recursive computation of $K_k$

\[
K_k (xa, x')
= \sum_{u \in A^k} \Phi_u (xa) \Phi_u (x')
= \sum_{u \in A^k} \Phi_u (x) \Phi_u (x') + \lambda \sum_{v \in A^{k-1}} \Psi_v (x) \Phi_v (x')
= K_k (x, x') + \lambda \sum_{v \in A^{k-1}} \Psi_v (x) \left( \sum_{j \in [1, |x'|]: x'_j = a} \Psi_v (x'_{[1,j-1]}) \lambda \right)
= K_k (x, x') + \lambda^2 \sum_{j \in [1, |x'|]: x'_j = a} B_{k-1} (x, x'_{[1,j-1]})
\]
Summary: Substring indexation

- Implementation in $O(|x| + |x'|)$ in memory and time for the spectrum and mismatch kernels (with suffix trees)
- Implementation in $O(k(|x| + |x'|))$ in memory and time for the spectrum and mismatch kernels (with tries)
- Implementation in $O(k|x| \times |x'|)$ 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 $\mathcal{D} = (x_1, x_2, \ldots, x_n)$
- Chose a **measure of similarity** $s(x, x')$
- Define the mapping $\Phi_{\mathcal{D}}(x) = (s(x, x_i))_{x_i \in \mathcal{D}}$
Dictionary-based indexation

The approach

1. Chose a **dictionary** of sequences $\mathcal{D} = (x_1, x_2, \ldots, x_n)$
2. Chose a **measure of similarity** $s(x, x')$
3. Define the mapping $\Phi_{\mathcal{D}}(x) = (s(x, x_i))_{x_i \in \mathcal{D}}$

Examples

This includes:

1. **Motif kernels** (Logan et al., 2001): the dictionary is a library of motifs, the similarity function is a matching function
2. **Pairwise kernel** (Liao & Noble, 2003): the dictionary is the training set, the similarity is a classical measure of similarity between sequences.
Outline

The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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 distributions

\[ \{P_\theta, \theta \in \Theta \subset \mathbb{R}^m\} \subset M^+_1(\mathcal{X}) \]
A context-tree model is a variable-memory Markov chain:

\[ P_{\mathcal{D},\theta}(x) = P_{\mathcal{D},\theta}(x_1 \ldots x_D) \prod_{i=D+1}^{n} P_{\mathcal{D},\theta}(x_i | x_{i-D} \ldots x_{i-1}) \]

- \( \mathcal{D} \) is a suffix tree
- \( \theta \in \Sigma^{\mathcal{D}} \) is a set of conditional probabilities (multinomials)
Context-tree model: example

\[ P(\text{AABACBACC}) = P(\text{AAB})\theta_{\text{AB}}(\text{A})\theta_{\text{A}}(\text{C})\theta_{\text{C}}(\text{B})\theta_{\text{ACB}}(\text{A})\theta_{\text{A}}(\text{C})\theta_{\text{C}}(\text{A}). \]
The context-tree kernel

Theorem (Cuturi et al., 2005)

For particular choices of priors, the context-tree kernel:

\[
K(x, x') = \sum_D \int_{\theta \in \Sigma^D} P_{D,\theta}(x) P_{D,\theta}(x') w(d\theta|D) \pi(D)
\]

can be computed in \(O(|x| + |x'|)\) 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.
Recall: Definition

For any observed data $x \in \mathcal{X}$, let a latent variable $y \in \mathcal{Y}$ be associated probabilistically through a conditional probability $P_x(dy)$.

Let $K_Z$ be a kernel for the complete data $z = (x, y)$.

Then the following kernel is a valid kernel on $\mathcal{X}$, called a marginalized kernel (Tsuda et al., 2002):

$$K_{\mathcal{X}}(x, x') := E_{P_x(dy) \times P_{x'}(dy')} K_Z(z, z')$$

$$= \int \int K_Z((x, y), (x', y')) P_x(dy) P_{x'}(dy') .$$
Example: HMM for normal/biased coin toss

Normal ($N$) and biased ($B$) coins (not observed)

Observed output are 0/1 with probabilities:

\[
\begin{align*}
\pi(0|N) &= 1 - \pi(1|N) = 0.5, \\
\pi(0|B) &= 1 - \pi(1|B) = 0.2.
\end{align*}
\]

Example of realization (complete data):

```
NNNNNBBBBBBBBBBBBNNNNNNNNNNNBBBBBBB
1001011101111010010111001111011
```
1-spectrum kernel on complete data

- If both \( x \in A^* \) and \( y \in S^* \) were observed, we might rather use the 1-spectrum kernel on the complete data \( z = (x, y) \):

\[
K_z (z, z') = \sum_{(a,s) \in A \times S} n_{a,s} (z) n_{a,s} (z') ,
\]

where \( n_{a,s} (x, y) \) for \( a = 0, 1 \) and \( s = N, B \) is the number of occurrences of \( s \) in \( y \) which emit \( a \) in \( x \).

- Example:

\[
z = 1001011101110100101110011110111011, \\
z' = 0011010110011111011010111101100101,
\]

\[
K_z (z, z') = n_1 (z) n_1 (z') + n_1 (z) n_1 (z') + n_0 (z) n_0 (z') + n_0 (z) n_0 (z') \\
= 7 \times 15 + 13 \times 6 + 9 \times 12 + 2 \times 1 = 293.
\]
The marginalized kernel for observed data is:

\[ K_X(x, x') = \sum_{y, y' \in S^*} K_Z((x, y), (x', y')) \cdot P(y|x) \cdot P(y'|x') \]

\[ = \sum_{(a, s) \in A \times S} \Phi_{a,s}(x) \cdot \Phi_{a,s}(x') , \]

with

\[ \Phi_{a,s}(x) = \sum_{y \in S^*} P(y|x) \cdot n_{a,s}(x, y) \]
Computation of the 1-spectrum marginalized kernel

\[ \Phi_{a,s}(x) = \sum_{y \in S^*} P(y|x) n_{a,s}(x, y) \]

\[ = \sum_{y \in S^*} P(y|x) \left\{ \sum_{i=1}^{n} \delta(x_i, a) \delta(y_i, s) \right\} \]

\[ = \sum_{i=1}^{n} \delta(x_i, a) \left\{ \sum_{y \in S^*} P(y|x) \delta(y_i, s) \right\} \]

\[ = \sum_{i=1}^{n} \delta(x_i, a) P(y_i = s|x). \]

and \( P(y_i = s|x) \) 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

The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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:

\[ \text{CGGSLIAMM------WFGV} \]
\[ | \ldots | | | | | | | | | \]
\[ C------LIVMMNRLMWFGV \]
In order to quantify the relevance of an alignment $\pi$, define:

- a substitution matrix $S \in \mathbb{R}^{A \times A}$
- a gap penalty function $g : \mathbb{N} \rightarrow \mathbb{R}$

Any alignment is then scored as follows:

$$s_{S,g}(\pi) = S(C, C) + S(L, L) + S(I, I) + S(A, V) + 2S(M, M) + S(W, W) + S(F, F) + S(G, G) + S(V, V) - g(3) - g(4)$$
Local alignment kernel

Smith-Waterman score (Smith and Waterman, 1981)

- The widely-used Smith-Waterman local alignment score is defined by:

\[
SW_{S,g}(x, y) := \max_{\pi \in \Pi(x,y)} s_{S,g}(\pi).
\]

- 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:

\[ SW_{S,g}(x, y) := \max_{\pi \in \Pi(x, y)} s_{S,g}(\pi). \]

- It is symmetric, but not positive definite...

LA kernel (Saigo et al., 2004)

The local alignment kernel:

\[ K^{(\beta)}_{LA}(x, y) = \sum_{\pi \in \Pi(x, y)} \exp(\beta s_{S,g}(x, y, \pi)), \]

is symmetric positive definite.
Lemma

- If $K_1$ and $K_2$ are p.d. kernels, then:
  
  $$K_1 + K_2,$$
  
  $$K_1 K_2,$$
  
  and
  
  $$cK_1,$$ 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 $K$:

  $$\forall (x, x') \in X^2, \quad K(x, x') = \lim_{n \to \infty} K_i(x, x'),$$

  then $K$ is also a p.d. kernel.
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 $f_1, \ldots, f_n$. 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. □
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,**

  $$K((x_1, x_2), (y_1, y_2)) = K_1(x_1, y_1) + K_2(x_2, y_2),$$

- **The direct product:**

  $$K((x_1, x_2), (y_1, y_2)) = K_1(x_1, y_1) K_2(x_2, y_2).$$
Proof of lemma

If $K_1$ is a p.d. kernel, let $\Phi_1 : \mathcal{X}_1 \rightarrow \mathcal{H}$ be such that:

$$K_1 (x_1, y_1) = \langle \Phi_1 (x_1), \Phi_1 (y_1) \rangle_{\mathcal{H}}.$$ 

Let $\Phi : \mathcal{X}_1 \times \mathcal{X}_2 \rightarrow \mathcal{H}$ be defined by:

$$\Phi ((x_1, x_2)) = \Phi_1 (x_1).$$

Then for $x = (x_1, x_2)$ and $y = (y_1, y_2) \in \mathcal{X}$, we get

$$\langle \Phi ((x_1, x_2)), \Phi ((y_1, y_2)) \rangle_{\mathcal{H}} = K_1 (x_1, x_2),$$

which shows that $K (x, y) := K_1 (x_1, y_1)$ 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. $\square$
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 \( K_P \) on \( \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \) defined by:

\[
\forall A, B \in \mathcal{P}(\mathcal{X}), \quad K_P(A, B) := \sum_{x \in A} \sum_{y \in B} K(x, y)
\]

is a p.d. kernel on \( \mathcal{P}(\mathcal{X}) \).
Proof of lemma

Let $\Phi : \mathcal{X} \mapsto \mathcal{H}$ be such that

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

Then, for $A, B \in \mathcal{P}(\mathcal{X})$, we get:

$$K_P(A, B) = \sum_{x \in A} \sum_{y \in B} \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}}$$

$$= \left\langle \sum_{x \in A} \Phi(x), \sum_{y \in B} \Phi(y) \right\rangle_{\mathcal{H}}$$

$$= \langle \Phi_P(A), \Phi_P(B) \rangle_{\mathcal{H}},$$

with $\Phi_P(A) := \sum_{x \in A} \Phi(x)$. □
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 $K_1 \ast K_2$, is defined for any $x, x' \in \mathcal{X}$ by:

$$K_1 \ast K_2(x, y) := \sum_{x_1x_2=x, y_1y_2=y} K_1(x_1, y_1)K_2(x_2, y_2).$$

Lemma

If $K_1$ and $K_2$ are p.d. then $K_1 \ast K_2$ is p.d..
Proof of lemma

Let $\mathcal{X}$ be the set of finite-length strings. For $x \in \mathcal{X}$, let

$$R(x) = \{(x_1, x_2) \in \mathcal{X} \times \mathcal{X} : x = x_1x_2\} \subset \mathcal{X} \times \mathcal{X}.$$ 

We can then write

$$K_1 \ast K_2(x, y) = \sum_{(x_1, x_2) \in R(x)} \sum_{(y_1, y_2) \in R(y)} K_1(x_1, y_1)K_2(x_2, y_2)$$

which is a p.d. kernel by the previous lemmas. $\square$
3 basic string kernels

- The constant kernel:
  \[ K_0(x, y) := 1. \]

- A kernel for letters:
  \[ K_a^{(\beta)}(x, y) := \begin{cases} 
  0 & \text{if } |x| \neq 1 \text{ where } |y| \neq 1, \\
  \exp(\beta S(x, y)) & \text{otherwise}. 
\end{cases} \]

- A kernel for gaps:
  \[ K_g^{(\beta)}(x, y) = \exp[\beta (g(|x|) + g(|y|))]. \]
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:

$$
K_g^{(\beta)}(x, y) = \exp(\beta g(|x|)) \times \exp(\beta g(|y|)).
$$
Lemma

The local alignment kernel is a (limit) of convolution kernel:

\[
K_{LA}^{(\beta)} = \sum_{n=0}^{\infty} K_0 \ast \left( K_a^{(\beta)} \ast K_g^{(\beta)} \right)^{(n-1)} \ast K_a^{(\beta)} \ast K_0.
\]

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:
  \[
  \begin{cases}
    g(0) &= 0, \\
    g(n) &= d + e(n - 1) \text{ if } n \geq 1,
  \end{cases}
  \]

- The LA kernel can then be computed by dynamic programming by:
  \[
  K^{(\beta)}_{LA}(x, y) = 1 + X_2(|x|, |y|) + Y_2(|x|, |y|) + M(|x|, |y|),
  \]

  where \(M(i, j), X(i, j), Y(i, j), X_2(i, j), \) and \(Y_2(i, j)\) for \(0 \leq i \leq |x|, \)
  and \(0 \leq j \leq |y|\) are defined recursively.
LA kernel is p.d.: proof (/)

Initialization

\[
\begin{align*}
M(i, 0) &= M(0, j) = 0, \\
X(i, 0) &= X(0, j) = 0, \\
Y(i, 0) &= Y(0, j) = 0, \\
X_2(i, 0) &= X_2(0, j) = 0, \\
Y_2(i, 0) &= Y_2(0, j) = 0,
\end{align*}
\]
Recursion

For $i = 1, \ldots, |x|$ and $j = 1, \ldots, |y|:

\[
\begin{align*}
M(i,j) &= \exp(\beta S(x_i, y_j)) \left[1 + X(i-1, j-1) + Y(i-1, j-1) + M(i-1, j-1)\right], \\
X(i,j) &= \exp(\beta d) M(i-1, j) + \exp(\beta e) X(i-1, j), \\
Y(i,j) &= \exp(\beta d) \left[M(i, j-1) + X(i, j-1)\right] + \exp(\beta e) Y(i, j-1), \\
X_2(i,j) &= M(i-1, j) + X_2(i-1, j), \\
Y_2(i,j) &= M(i, j-1) + X_2(i, j-1) + Y_2(i, j-1).
\end{align*}
\]
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

The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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** cannot be detected by direct sequence comparison
SCOP database

- SCOP
- Fold
- Superfamily
- Family

Remote homologs
Close homologs
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.
Performance on the SCOP superfamily recognition benchmark (from Saigo et al., 2004).
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. Kernel tricks
3. Kernel Methods: Supervised Learning
5. The Kernel Jungle
   - Green, Mercer, Herglotz, Bochner and friends
   - Kernels for probabilistic models
   - Kernels for biological sequences
   - Kernels for graphs
   - Kernels on graphs
6. Open Problems and Research Topics
The Kernel Jungle

Green, Mercer, Herglotz, Bochner and friends
Kernels for probabilistic models
Kernels for biological sequences
Kernels for graphs
  Motivation
  Explicit enumeration of features
  Challenges
  Walk-based kernels
  Applications
Kernels on graphs
Virtual screening for drug discovery

Image retrieval and classification

From Harchaoui and Bach (2007).
Our approach

Represent each graph $x$ in $X$ by a vector $\Phi(x) \in H$, either explicitly or implicitly through the kernel $K(x, x') = \Phi(x) \top \Phi(x')$.

Use a linear method for classification in $H$. 
Our approach

1. Represent each graph $x$ in $\mathcal{X}$ by a vector $\Phi(x) \in \mathcal{H}$, either explicitly or implicitly through the kernel

$$K(x, x') = \Phi(x)^\top \Phi(x').$$
Our approach

1. Represent each graph $x$ in $\mathcal{X}$ by a vector $\Phi(x) \in \mathcal{H}$, either explicitly or implicitly through the kernel

$$K(x, x') = \Phi(x)^\top \Phi(x').$$

2. Use a linear method for classification in $\mathcal{H}$. 
Outline

The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic models
- Kernels for biological sequences
- Kernels for graphs
  - Motivation
  - Explicit enumeration of features
  - Challenges
  - Walk-based kernels
  - Applications
- Kernels on graphs
The approach

1. Represent explicitly each graph $\mathbf{x}$ by a vector of fixed dimension $\Phi(\mathbf{x}) \in \mathbb{R}^p$. 

\[ \phi \]

\[ X \]

\[ \Phi \]

\[ H \]
The approach

1. Represent explicitly each graph $x$ by a vector of fixed dimension $\Phi(x) \in \mathbb{R}^p$.
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 predefined structures

- Use a machine learning algorithm such as SVM, kNN, PLS, decision tree, etc.
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
Often we believe that the presence or absence of particular substructures may be important predictive patterns. Hence it makes sense to represent a graph by features that indicate the presence (or the number of occurrences) of these substructures. However, detecting the presence of particular substructures may be computationally challenging...
Subgraphs

Definition

A subgraph of a graph \((V, E)\) is a graph \((V', E')\) with \(V' \subset V\) and \(E' \subset E\).

A graph and all its connected subgraphs.
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 a Hamiltonian path; the decision problem whether a graph has a Hamiltonian path is NP-complete.
Theorem

Computing all subgraph occurrences is \textit{NP-hard}.
Indexing by all subgraphs?

Theorem

Computing all subgraph occurrences is \textsc{NP-hard}.

Proof

- The linear graph of size $n$ is a subgraph of a graph $X$ with $n$ vertices iff $X$ has a Hamiltonian path;
- The decision problem whether a graph has a Hamiltonian path is \textsc{NP-complete}. 
A path of a graph \((V, E)\) is a sequence of distinct vertices \(v_1, \ldots, v_n \in V\) (\(i \neq j \implies v_i \neq v_j\)) such that \((v_i, v_{i+1}) \in E\) for \(i = 1, \ldots, n - 1\).

Equivalently the paths are the linear subgraphs.
Indexing by all paths?

(0, ..., 0, 1, 0, ..., 0, 1, 0, ...)

Theorem
Computing all path occurrences is NP-hard.

Proof
Same as for subgraphs.
Indexing by all paths?

\begin{align*}
\begin{tikzpicture}
  \node (A) at (0,0) {$A$};
  \node (B) at (1,0) {$B$};
  \node (C) at (1.5,.5) {$A$};
  \draw (A) -- (B);
  \draw (A) -- (C);
  \draw (C) -- (B);
\end{tikzpicture}
\end{align*}

\begin{align*}
(0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots)
\end{align*}

\begin{align*}
\begin{tikzpicture}
  \node (A) at (0,0) {$A$};
  \node (B) at (1,0) {$A$};
  \draw (A) -- (B);
\end{tikzpicture}
\quad
\begin{tikzpicture}
  \node (A) at (0,0) {$A$};
  \node (B) at (1,0) {$B$};
  \node (C) at (2,0) {$A$};
  \draw (A) -- (B);
  \draw (B) -- (C);
\end{tikzpicture}
\end{align*}

\textbf{Theorem}

Computing all path occurrences is \textit{NP-hard}. 
Indexing by all paths?

\[(0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots)\]

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 computationally efficient indexing (non-exhaustive list)

- substructures selected by domain knowledge (MDL fingerprint)
- all paths up to length $k$ (Openeye fingerprint, Nicholls 2005)
- all shortest path lengths (Borgwardt and Kriegel, 2005)
- all subgraphs up to $k$ vertices (graphlet kernel, Shervashidze et al., 2009)
- all frequent subgraphs in the database (Helma et al., 2004)
Example: Indexing by all shortest path lengths and their endpoint labels

$$\langle 0, \ldots, 0, 2, 0, \ldots, 0, 1, 0, \ldots \rangle$$
Example: Indexing by all shortest path lengths and their endpoint labels

Properties (Borgwardt and Kriegel, 2005)

- There are $O(n^2)$ shortest paths.
- The vector of counts can be computed in $O(n^3)$ with the Floyd-Warshall algorithm.
Example: Indexing by all subgraphs up to $k$ vertices

$\begin{array}{c}
\text{A} \quad \text{A} \\
\text{B} \quad \text{A} \\
\end{array} \quad (0, \ldots, 0,1,0,\ldots, 0,1,0,\ldots)$

$\begin{array}{c}
\text{A} \\
\end{array}$

$\begin{array}{c}
\text{A} \\
\text{B} \\
\text{A} \\
\end{array}$
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.
Explicit computation of substructure occurrences can be computationally prohibitive (subgraphs, 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

5 The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic models
- Kernels for biological sequences
- Kernels for graphs
  - Motivation
  - Explicit enumeration of features
  - Challenges
    - Walk-based kernels
    - Applications
- Kernels on graphs
The idea

Represent implicitly each graph $x$ in $X$ by a vector $\Phi(x) \in H$ through the kernel $K(x, x') = \Phi(x)\top\Phi(x')$.

Use a kernel method for classification in $H$. 
The idea

1. Represent implicitly each graph \( x \) in \( \mathcal{X} \) by a vector \( \Phi(x) \in \mathcal{H} \) through the kernel

\[
K(x, x') = \Phi(x)^\top \Phi(x').
\]
The idea

1. Represent implicitly each graph $x$ in $\mathcal{X}$ by a vector $\Phi(x) \in \mathcal{H}$ through the kernel

$$K(x, x') = \Phi(x)^\top \Phi(x').$$

2. Use a kernel method for classification in $\mathcal{H}$. 
Expressiveness vs Complexity

Definition: Complete graph kernels

A graph kernel is complete if it distinguishes 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 distinguishes 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

<table>
<thead>
<tr>
<th>Proposition (Gärtner et al., 2003)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing <strong>any complete graph kernel</strong> is at least as hard as the graph isomorphism problem.</td>
</tr>
</tbody>
</table>
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 $G_1 \cong G_2$).
Subgraph kernel

**Definition**
- Let $\{\lambda_G\}_{G \in \mathcal{X}}$ be a set of nonnegative real-valued weights.
- For any graph $G \in \mathcal{X}$ and any connected graph $H \in \mathcal{X}$, let
  \[
  \Phi_H(G) = \left| \{ G' \text{ is a subgraph of } G : G' \cong H \} \right| .
  \]

- The **subgraph kernel** between any two graphs $G_1$ and $G_2 \in \mathcal{X}$ is defined by:
  \[
  K_{\text{subgraph}}(G_1, G_2) = \sum_{\substack{H \in \mathcal{X} \\ H \text{ connected}}} \lambda_H \Phi_H(G_1) \Phi_H(G_2).
  \]
Subgraph kernel complexity

Proposition (Gärtner et al., 2003)
Computing the subgraph kernel is NP-hard.

Proof (1/2)
Let $P_n$ be the path graph with $n$ vertices.
Subgraphs of $P_n$ are path graphs:
$$\Phi(P_n) = e_{P_1} + (n-1)e_{P_2} + \ldots + e_{P_n}.$$ 

The vectors $\Phi(P_1), \ldots, \Phi(P_n)$ 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 an $n \times n$ triangular system).
Subgraph kernel complexity

Proposition (Gärtner et al., 2003)
Computing the subgraph kernel is \textbf{NP-hard}.

Proof (1/2)

- Let $P_n$ be the path graph with $n$ vertices.
- Subgraphs of $P_n$ are path graphs:
  \[
  \Phi(P_n) = ne_{P_1} + (n-1)e_{P_2} + \ldots + e_{P_n}.
  \]
- The vectors $\Phi(P_1), \ldots, \Phi(P_n)$ 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 an $n \times n$ triangular system).
Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is \textbf{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 $\Phi(G)^\top e_{P_n} > 0$, i.e.,

$$
\Phi(G)^\top \left( \sum_{i=1}^{n} \alpha_i \Phi(P_i) \right) = \sum_{i=1}^{n} \alpha_i K_{\text{subgraph}}(G, P_i) > 0.
$$

- The decision problem whether a graph has a Hamiltonian path is \textbf{NP-complete}. \qed
Path kernel

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 \( \mathcal{P} \subseteq \mathcal{X} \) is the set of path graphs.
**Path kernel**

The path kernel is the subgraph kernel restricted to paths, i.e.,

\[ K_{\text{path}}(G_1, G_2) = \sum_{H \in P} \lambda_H \Phi_H(G_1) \Phi_H(G_2), \]

where \( P \subset X \) is the set of path graphs.

**Proposition (Gärtner et al., 2003)**

Computing the path kernel is \text{NP-hard}. 

\[ (0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots) \]
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

The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic models
- Kernels for biological sequences

Kernels for graphs

- Motivation
- Explicit enumeration of features
- Challenges
  - Walk-based kernels
    - Applications

Kernels on graphs
Walks

Definition

- A walk of a graph \((V, E)\) is a sequence of \(v_1, \ldots, v_n \in V\) such that \((v_i, v_{i+1}) \in E\) for \(i = 1, \ldots, n - 1\).

- We note \(W_n(G)\) the set of walks with \(n\) vertices of the graph \(G\), and \(W(G)\) the set of all walks.
Walks $\neq$ paths
Walk kernel

Definition

- Let $S_n$ denote the set of all possible label sequences of walks of length $n$ (including vertex and edge labels), and $S = \bigcup_{n \geq 1} S_n$.
- 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 $\Phi(G) = (\Phi_s(G))_{s \in S}$ be defined by:

$$
\Phi_s(G) = \sum_{w \in \mathcal{W}(G)} \lambda_G(w) \mathbf{1} \left( s \text{ is the label sequence of } w \right).
$$
Walk kernel

Definition

- Let $S_n$ denote the set of all possible label sequences of walks of length $n$ (including vertex and edge labels), and $S = \bigcup_{n \geq 1} S_n$.
- 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 $\Phi(G) = (\Phi_s(G))_{s \in S}$ be defined by:
  \[ \Phi_s(G) = \sum_{w \in \mathcal{W}(G)} \lambda_G(w) 1 \text{(s is the label sequence of w)}. \]
- A walk kernel is a graph kernel defined by:
  \[ K_{walk}(G_1, G_2) = \sum_{s \in S} \Phi_s(G_1) \Phi_s(G_2). \]
Walk kernel examples

Examples

- The \textit{nth-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 \textit{nth-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 \textit{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:

$$K(G_1, G_2) = P(label(W_1) = label(W_2)),$$

where $W_1$ and $W_2$ are two independent random walks on $G_1$ and $G_2$, respectively (Kashima et al., 2003).

- The \textit{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 (\textit{nth-order, random and geometric walk kernels}) can be computed efficiently in \textit{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. $V = \{(v_1, v_2) \in V_1 \times V_2 : v_1$ and $v_2$ have the same label\}$,
2. $E = \{((v_1, v_2), (v'_1, v'_2)) \in V \times V : (v_1, v'_1) \in E_1$ and $(v_2, v'_2) \in E_2\}$. 

![Diagram](image.png)
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

$$K_{walk}(G_1, G_2) = \sum_{s \in S} \Phi_s(G_1)\Phi_s(G_2)$$

$$= \sum_{(w_1, w_2) \in \mathcal{W}(G_1) \times \mathcal{W}(G_1)} \lambda_{G_1}(w_1)\lambda_{G_2}(w_2)1(l(w_1) = l(w_2))$$

$$= \sum_{w \in \mathcal{W}(G_1 \times G_2)} \lambda_{G_1 \times G_2}(w).$$
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:
  \[ K_{n\text{-th-order}}(G_1, G_2) = \sum_{w \in \mathcal{W}_n(G_1 \times G_2)} 1. \]

- Let $A$ be the adjacency matrix of $G_1 \times G_2$. Then we get:
  \[ K_{n\text{-th-order}}(G_1, G_2) = \sum_{i,j} [A^n]_{i,j} = 1^\top A^n 1. \]

- Computation in $O(n|V_1||V_2|d_1d_2)$, 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 $w = v_1 \ldots v_n$ can be decomposed as:

$$
\lambda_G(v_1 \ldots v_n) = \lambda_i(v_1) \prod_{i=2}^{n} \lambda^t(v_{i-1}, v_i).
$$

- Let $\Lambda_i$ be the vector of $\lambda^i(v)$ and $\Lambda_t$ be the matrix of $\lambda^t(v, v')$:

$$
K_{walk}(G_1, G_2) = \sum_{n=1}^{\infty} \sum_{w \in W_n(G_1 \times G_2)} \lambda^i(v_1) \prod_{i=2}^{n} \lambda^t(v_{i-1}, v_i)
$$

$$
= \sum_{n=0}^{\infty} \Lambda_i \Lambda_t^n 1
$$

$$
= \Lambda_i (I - \Lambda_t)^{-1} 1
$$

- Computation in $O(|V_1|^3 |V_2|^3)$. 
Compromise between fingerprints and structural keys.

Other relabeling schemes are possible.

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 $w = v_1 \ldots v_n$ 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**).
Remark: Here and in subsequent slides by *subtree* we mean a tree-like pattern with potentially repeated nodes and edges.
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 computing the (weighted) number of subtrees in the product graph.
- Recursion: if $T(v, n)$ denotes the weighted number of subtrees of depth $n$ rooted at the vertex $v$, then:
  \[
  T(v, n + 1) = \sum_{R \subseteq N(v)} \prod_{v' \in R} \lambda_t(v, v') T(v', n),
  \]
  where $N(v)$ 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.
Back to label enrichment

Link between the Morgan index and subtrees

Recall the Morgan index:

The Morgan index of order $k$ at a node $v$ in fact corresponds to the number of leaves in the $k$-th order full subtree pattern rooted at $v$.

A full subtree pattern of order 2 rooted at node 1.
Label enrichment via the Weisfeiler-Lehman algorithm

A slightly more involved label enrichment strategy (Weisfeiler and Lehman, 1968) is exploited in the definition and computation of the Weisfeiler-Lehman subtree kernel (Shervashidze and Borgwardt, 2009).

1. Multiset-label determination and sorting

2. Label compression

3. Relabeling

Compressed labels represent full subtree patterns.
Label enrichment via the Weisfeiler-Lehman algorithm

A slightly more involved label enrichment strategy (Weisfeiler and Lehman, 1968) is exploited in the definition and computation of the Weisfeiler-Lehman subtree kernel (Shervashidze and Borgwardt, 2009).

1. Multiset-label determination and sorting

2. Label compression

3. Relabeling

Compressed labels represent full subtree patterns.
Weisfeiler-Lehman (WL) subtree kernel

**Properties**

- The WL features up to the $k$-th order are computed in $O(|E|k)$.
- Similarly to the Morgan index, the WL relabeling can be exploited in combination with any graph kernel (that takes into account categorical node labels) to make it more expressive (Shervashidze et al., 2011).

\[
\varphi_{WL_{subtree}}^{(1)}(G) = (2, 1, 1, 1, 1, 2, 0, 1, 0, 1, 1, 0, 1)
\]

\[
\varphi_{WL_{subtree}}^{(1)}(G') = (1, 2, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1)
\]
Outline

The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic models
- Kernels for biological sequences
- Kernels for graphs
  - Motivation
  - Explicit enumeration of features
  - 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Progol1</td>
<td>81.4%</td>
</tr>
<tr>
<td>2D kernel</td>
<td>91.2%</td>
</tr>
</tbody>
</table>
2D subtree vs walk kernels

Screening of inhibitors for 60 cancer cell lines.
Comparison of several graph feature extraction methods/kernels (Shervashidze et al., 2011)

10-fold cross-validation accuracy on graph classification problems in chemo- and bioinformatics:

- NCI1 and NCI109 - active/inactive compounds in an anti-cancer screen
- ENZYMES - 6 types of enzymes from the BRENDA database

<table>
<thead>
<tr>
<th>Method/Data Set</th>
<th>NCI1</th>
<th>NCI109</th>
<th>ENZYMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>WL subtree</td>
<td>82.19 (±0.18)</td>
<td>82.46 (±0.24)</td>
<td>52.22 (±1.26)</td>
</tr>
<tr>
<td>WL shortest path</td>
<td>84.55 (±0.36)</td>
<td>83.53 (±0.30)</td>
<td>59.05 (±1.05)</td>
</tr>
<tr>
<td>Ramon &amp; Gärtner</td>
<td>61.86 (±0.27)</td>
<td>61.67 (±0.21)</td>
<td>13.35 (±0.87)</td>
</tr>
<tr>
<td>Geometric p-walk</td>
<td>58.66 (±0.28)</td>
<td>58.36 (±0.94)</td>
<td>27.67 (±0.95)</td>
</tr>
<tr>
<td>Geometric walk</td>
<td>64.34 (±0.27)</td>
<td>63.51 (±0.18)</td>
<td>21.68 (±0.94)</td>
</tr>
<tr>
<td>Graphlet count</td>
<td>66.00 (±0.07)</td>
<td>66.59 (±0.08)</td>
<td>32.70 (±1.20)</td>
</tr>
<tr>
<td>Shortest path</td>
<td>73.47 (±0.11)</td>
<td>73.07 (±0.11)</td>
<td>41.68 (±1.79)</td>
</tr>
</tbody>
</table>
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).

![Image examples](image1.png) ![Image examples](image2.png) ![Image examples](image3.png)

Performance comparison on Corel14

<table>
<thead>
<tr>
<th>Kernels</th>
<th>Test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>0.05</td>
</tr>
<tr>
<td>W</td>
<td>0.06</td>
</tr>
<tr>
<td>TW</td>
<td>0.07</td>
</tr>
<tr>
<td>wTW</td>
<td>0.08</td>
</tr>
<tr>
<td>M</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Graph showing performance comparison with box plots for each kernel type.
**Summary: graph kernels**

<table>
<thead>
<tr>
<th>What we saw</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Kernels do <strong>not allow</strong> to overcome the NP-hardness of subgraph patterns.</td>
</tr>
<tr>
<td>- They allow to work with approximate subgraphs (walks, subtrees) in infinite dimension, thanks to the <strong>kernel trick</strong>.</td>
</tr>
<tr>
<td>- However: using kernels makes it difficult to <strong>come back to patterns</strong> after the learning stage.</td>
</tr>
</tbody>
</table>
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
5. The Kernel Jungle
   - Green, Mercer, Herglotz, Bochner and friends
   - Kernels for probabilistic models
   - Kernels for biological sequences
   - Kernels for graphs
   - Kernels on graphs
6. Open Problems and Research Topics
Outline

The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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

Data often 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
We need a kernel $K(x, x')$ between nodes of the graph.

Example: predict protein functions from high-throughput protein-protein interaction data.
General remarks

Strategies to design 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**

<table>
<thead>
<tr>
<th>Strategies to design a kernel on a graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>• $\mathcal{X}$ being finite, any symmetric semi-definite matrix $K$ defines a valid p.d. kernel on $\mathcal{X}$.</td>
</tr>
<tr>
<td>• How to “translate” the graph topology into the kernel?</td>
</tr>
<tr>
<td>• <strong>Direct geometric approach:</strong> $K_{i,j}$ should be “large” when $x_i$ and $x_j$ are “close” to each other on the graph?</td>
</tr>
<tr>
<td>• <strong>Functional approach:</strong> $|f|_K$ should be “small” when $f$ is “smooth” on the graph?</td>
</tr>
<tr>
<td>• <strong>Link discrete/continuous:</strong> is there an equivalent to the continuous Gaussian kernel on the graph (e.g., limit by fine discretization)?</td>
</tr>
</tbody>
</table>
The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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. kernel is an inner product in a Hilbert space
  \[ K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}. \]

- It defines a Hilbert distance:
  \[ d_K^2(x, x') = K(x, x) + K(x', x') - 2K(x, x'). \]

- \(-d_K^2\) is conditionally positive definite (c.p.d.), i.e.:
  \[ \forall t > 0, \quad \exp \left( -td_K(x, x')^2 \right) \text{ is p.d.} \]
Example

A direct approach

- For $\mathcal{X} = \mathbb{R}^n$, the inner product is p.d.:

$$K(x, x') = x^\top x'.$$

- The corresponding Hilbert distance is the Euclidean distance:

$$d_K(x, x')^2 = x^\top x + x'^\top x' - 2x^\top x' = ||x - x'||^2.$$

- $-d_K^2$ is conditionally positive definite (c.p.d.), i.e.:

$$\forall t > 0, \quad \exp (-t||x - x'||^2) \text{ is p.d.}$$
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 $-d_G$ 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 $-d_G$ 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 **cannot** be embedded exactly in Hilbert spaces.
- **In some cases** exact embeddings exist, e.g.:
  - *trees* can be embedded exactly,
  - *closed chains* can be embedded exactly.
Example: non-c.p.d. graph distance

\[
d_G = \begin{pmatrix}
0 & 1 & 1 & 1 & 2 \\
1 & 0 & 2 & 2 & 1 \\
1 & 2 & 0 & 2 & 1 \\
1 & 2 & 2 & 0 & 1 \\
2 & 1 & 1 & 1 & 0
\end{pmatrix}
\]

\[
\lambda_{\min} \left( e^{-0.2d_G(i,j)} \right) = -0.028 < 0.
\]
Graph distances on trees are c.p.d.

Proof

- Let $G = (V, E)$ be 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 part of 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

\[
\begin{bmatrix}
  e^{-d_G(i,j)}
\end{bmatrix}
= 
\begin{pmatrix}
  1 & 0.14 & 0.37 & 0.14 & 0.05 \\
  0.14 & 1 & 0.37 & 0.14 & 0.05 \\
  0.37 & 0.37 & 1 & 0.37 & 0.14 \\
  0.14 & 0.14 & 0.37 & 1 & 0.37 \\
  0.05 & 0.05 & 0.14 & 0.37 & 1
\end{pmatrix}
\]
Graph distances on closed chains are c.p.d.

Proof: case $|V| = 2p$

- Let $G = (V, E)$ be a directed cycle with an even number of vertices $|V| = 2p$.
- Fix a root $x_0 \in V$, number the $2p$ edges from $x_0$ to $x_0$;
- Label the $2p$ edges with $e_1, \ldots, e_p, -e_1, \ldots, -e_p$ (vectors in $\mathbb{R}^p$);
- For a vertex $v$, take $\Phi(v)$ to be the sum of the labels of the edges in the shortest directed path between $x_0$ and $v$. 

\[ \begin{align*}
\text{Diagram 1} & \quad \text{Diagram 2}
\end{align*} \]
Another interesting graph

- Let $\mathcal{S}_n$ the set of permutations of $n$ items (symmetric group)
- Cayley graph $G$: connect two permutations when they differ by one adjacent transposition
- $d_G$ can be computed in $O(n \log n)$
  \textit{how?}
- $d_G$ is c.p.d. \textit{why?}
- See Jiao and Vert (2017)
Some graph distances are c.p.d, some are not
There is a large literature in mathematics on how to "approximately" embed a graph; maybe this could be useful for machine learning?
Graph distance is very sensitive to "noise" in edges
We need other approaches to define a p.d. kernel that would work for all graphs, and be less sensitive to noise in the edges.
Outline

The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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 design a p.d. kernel on general graphs?
- Designing a kernel is equivalent to defining an 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 an RKHS and identify the corresponding kernel.
Notations

- \( \mathcal{X} = (x_1, \ldots, x_m) \) is finite.
- For \( x, x' \in \mathcal{X} \), we note \( x \sim x' \) to indicate the existence of an edge between \( x \) and \( x' \).
- We assume that there is no self-loop \( x \sim x \), 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 \( x_i \) \((D_{i,i} = \sum_{j=1}^{m} A_{i,j})\).
Example

\[ A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \]
Graph Laplacian

Definition

The Laplacian of the graph is the matrix $L = D - A$. 

$L = D - A = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$
Properties of the Laplacian

**Lemma**

Let $L = D - A$ be the Laplacian of a connected graph:

- For any $f : \mathcal{X} \to \mathbb{R}$,
  $$\Omega(f) := \sum_{i \sim j} (f(x_i) - f(x_j))^2 = f^\top Lf$$

- $L$ is a **symmetric positive semi-definite matrix**

- 0 is an **eigenvalue** with multiplicity 1 associated to the constant eigenvector $1 = (1, \ldots, 1)$

- 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 $L$

$$
\Omega(f) = \sum_{i \sim j} (f(x_i) - f(x_j))^2
= \sum_{i \sim j} \left( f(x_i)^2 + f(x_j)^2 - 2f(x_i)f(x_j) \right)
= \sum_{i=1}^{m} D_{i,i} f(x_i)^2 - 2 \sum_{i \sim j} f(x_i) f(x_j)
= f^T Df - f^T Af
= f^T Lf
$$
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 $\sum_{i \sim j} (f(x_i) - f(x_j))^2 = 0$
  - iff $f(x_i) = f(x_j)$ 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 $1$. □
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

\[
\Omega(f) = \sum_{i \sim j} (f(x_i) - f(x_j))^2
\]

is a RKHS whose reproducing kernel is \( L^* \), the pseudo-inverse of the graph Laplacian.
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:

$$\langle f, g \rangle = f^\top L g$$

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:

$$\| f \|^2 = \langle f, f \rangle = f^\top L f = \Omega(f) .$$
Proof (2/2)

To check that $\mathcal{H}$ is a RKHS with reproducing kernel $K = L^*$, it suffices to show that:

$$
\begin{align*}
\forall x \in \mathcal{X}, & \quad K_x \in \mathcal{H}, \\
\forall (x, f) \in \mathcal{X} \times \mathcal{H}, & \quad \langle f, K_x \rangle = f(x).
\end{align*}
$$

- $\text{Ker}(K) = \text{Ker}(L^*) = \text{Ker}(L)$, implying $K1 = 0$. 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 $\mathcal{H}$. □
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

\[ \Delta f(x) = f''''(x) \]

\[ \sim \frac{f'(x + dx/2) - f'(x - dx/2)}{dx} \]

\[ \sim \frac{f(x + dx) - f(x) - f(x) + f(x - dx)}{dx^2} \]

\[ = \frac{f_{i-1} + f_{i+1} - 2f_i}{dx^2} \]

\[ = -\frac{Lf(i)}{dx^2}. \]
Interpretation of regularization

For $f = [0, 1] \rightarrow \mathbb{R}$ and $x_i = i/m$, we have:

$$
\Omega(f) = \sum_{i=1}^{m} \left( f \left( \frac{i + 1}{m} \right) - f \left( \frac{i}{m} \right) \right)^2 \\
\sim \sum_{i=1}^{m} \left( \frac{1}{m} \times f' \left( \frac{i}{m} \right) \right)^2 \\
= \frac{1}{m} \times \frac{1}{m} \sum_{i=1}^{m} f' \left( \frac{i}{m} \right)^2 \\
\sim \frac{1}{m} \int_{0}^{1} f'(t)^2 dt.
$$
Outline

The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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 $\mathbb{R}^d$:

$$K_t (x, x') = \frac{1}{(4\pi t)^{d/2}} \exp \left( -\frac{\|x - x'\|^2}{4t} \right).$$

In order to transpose it to the graph, replacing the Euclidean distant 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 $x_0 \in \mathbb{R}^d$, the function:

$$K_{x_0}(x, t) = K_t(x_0, x) = \frac{1}{(4\pi t)^{d/2}} \exp\left(-\frac{\|x - x_0\|^2}{4t}\right)$$

is solution of the diffusion equation:

$$\frac{\partial}{\partial t} K_{x_0}(x, t) = \Delta K_{x_0}(x, t)$$

with initial condition $K_{x_0}(x, 0) = \delta_{x_0}(x)$

(proof by direct computation).
For finite-dimensional $f_t \in \mathbb{R}^m$, the diffusion equation becomes:

$$\frac{\partial}{\partial t} f_t = -Lf_t$$

which admits the following solution:

$$f_t = f_0 e^{-tL}$$

with

$$e^{tL} = I - tL + \frac{t^2}{2!}L^2 - \frac{t^3}{3!}L^3 + \ldots$$
Diffusion kernel (Kondor and Lafferty, 2002)

This suggest to consider:

$$K = e^{-tL}$$

which is indeed symmetric positive semi-definite because if we write:

$$L = \sum_{i=1}^{m} \lambda_i u_i u_i^\top \quad (\lambda_i \geq 0)$$

we obtain:

$$K = e^{-tL} = \sum_{i=1}^{m} e^{-t\lambda_i} u_i u_i^\top$$
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

\[
K_{i,j} = \frac{1}{m} \sum_{\nu=0}^{m-1} \exp \left[ -2t \left( 1 - \cos \frac{2\pi \nu}{m} \right) \right] \cos \frac{2\pi \nu (i-j)}{m}.
\]
The Kernel Jungle
- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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 $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_m$ 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 $f = K \left(K^{-1}f\right)$, therefore its norm in the diffusion RKHS is:

\[
\| f \|_{K_t}^2 = \left(f^\top K^{-1}\right) K \left(K^{-1}f\right) = f^\top K^{-1}f.
\]

- For $i = 1, \ldots, m$, 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 $\int \left| \hat{f}(\omega) \right|^2 e^{\sigma^2\omega^2} d\omega$...
Discrete Fourier transform

**Definition**

The vector \( \hat{f} = (\hat{f}_1, \ldots, \hat{f}_m)^\top \) 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 \( \lambda_i \) are the equivalent to the frequencies \( \omega^2 \).
- Successive eigenvectors “oscillate” increasingly as eigenvalues get more and more negative.
Examples

\( \lambda = 0.12 \)

\( \Lambda = 0.76 \)
Examples

\[ \lambda = 0.47 \]

\[ \text{Lambda} = 0.83 \]
Examples

\[ \lambda = 1 \]

\[ \Lambda = 1.3 \]
Examples

\[ \text{lambda} = 1.7 \]

\[ \text{Lambda} = 2.2 \]
Examples

\[ \text{lambda} = 2.3 \]

\[ \text{Lambda} = 2.8 \]
Examples

\begin{align*}
\lambda &= 3 \\
\Lambda &= 3.6
\end{align*}
Examples

\[ \lambda = 3.5 \]

\[ \text{Lambda} = 4.2 \]
Examples

\[
\text{lambda} = 3.9
\]

\[
\text{Lambda} = 6.3
\]
Generalization

This observation suggests to define a whole family of kernels:

\[ K_r = \sum_{i=1}^{m} r(\lambda_i)u_iu_i^\top \]

associated with the following RKHS norms:

\[ \| f \|_{K_r}^2 = \sum_{i=1}^{m} \frac{\hat{f}_i^2}{r(\lambda_i)} \]

where \( r : \mathbb{R}^+ \to \mathbb{R}^*_+ \) is a non-increasing function.
Example: regularized Laplacian

\[ r(\lambda) = \frac{1}{\lambda + \epsilon}, \quad \epsilon > 0 \]

\[ K = \sum_{i=1}^{m} \frac{1}{\lambda_i + \epsilon} u_i u_i^\top = (L + \epsilon I)^{-1} \]

\[ \| f \|^2_K = f^\top K^{-1} f = \sum_{i \sim j} (f(x_i) - f(x_j))^2 + \epsilon \sum_{i=1}^{m} f(x_i)^2. \]
Example

\[(L + I)^{-1} = \begin{pmatrix}
0.60 & 0.10 & 0.19 & 0.08 & 0.04 \\
0.10 & 0.60 & 0.19 & 0.08 & 0.04 \\
0.19 & 0.19 & 0.38 & 0.15 & 0.08 \\
0.08 & 0.08 & 0.15 & 0.46 & 0.23 \\
0.04 & 0.04 & 0.08 & 0.23 & 0.62
\end{pmatrix}\]
The Kernel Jungle

- Green, Mercer, Herglotz, Bochner and friends
- Kernels for probabilistic 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:

\[
\min_{f \in \mathbb{R}^X} \sum_{i \sim j} (f_i - f_j)^2 \quad \text{s.t.} \quad \sum_i f_i^2 = 1
\]

- 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 $x_1, \ldots, x_q$ be a set of $q$ nodes (the query). How to find “similar” nodes (and rank them)?

One solution:

$$\min_{f} \| f \|_{\mathcal{H}} \quad \text{s.t.} \quad f(x_i) \geq 1 \text{ for } i = 1, \ldots, q.$$
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 \( x = (x_1, \ldots, x_p) \) 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!
Open Problems and Research Topics
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
5. The Kernel Jungle
6. Open Problems and Research Topics
   - Multiple Kernel Learning (MKL)
   - Large-scale learning with kernels
   - Foundations of deep learning from a kernel point of view
We have seen how to make learning algorithms given a kernel $K$ on some data space $\mathcal{X}$.

Often we may have several possible kernels:

- by varying the kernel type or parameters on a given description of the data (e.g., linear, polynomial, Gaussian kernels with different bandwidths...)
- because we have different views of the same data, e.g., a protein can be characterized by its sequence, its structure, its mass spectrometry profile...

How to choose or integrate different kernels in a learning task?
Setting: learning with one kernel

- For any \( f : \mathcal{X} \to \mathbb{R} \), let \( f^n = (f(x_1), \ldots, f(x_n)) \in \mathbb{R}^n \)
- Given a p.d. kernel \( K : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \), we learn with \( K \) by solving:

\[
\min_{f \in \mathcal{H}} R(f^n) + \lambda \| f \|_\mathcal{H}^2 ,
\]

where \( \lambda > 0 \) and \( R : \mathbb{R}^n \to \mathbb{R} \) is an closed\(^3\) and convex empirical risk:

- \( R(u) = \frac{1}{n} \sum_{i=1}^{n} (u_i - y_i)^2 \) for kernel ridge regression
- \( R(u) = \frac{1}{n} \sum_{i=1}^{n} \max(1 - y_i u_i, 0) \) for SVM
- \( R(u) = \frac{1}{n} \sum_{i=1}^{n} \log (1 + \exp (-y_i u_i)) \) for kernel logistic regression

---

\(^3\) \( R \) is closed if, for each \( A \in \mathbb{R} \), the sublevel set \( \{ u \in \mathbb{R}^n : R(u) \leq A \} \) is closed. For example, if \( R \) is continuous then it is closed.
Sum kernel

Definition

Let $K_1, \ldots, K_M$ be $M$ kernels on $\mathcal{X}$. The sum kernel $K_S$ is the kernel on $\mathcal{X}$ defined as

$$\forall x, x' \in \mathcal{X}, \quad K_S(x, x') = \sum_{i=1}^{M} K_i(x, x').$$
Theorem

For \( i = 1, \ldots, M \), let \( \Phi_i : X \rightarrow \mathcal{H}_i \) be a feature map such that

\[
K_i(x, x') = \langle \Phi_i(x), \Phi_i(x') \rangle_{\mathcal{H}_i} .
\]

Then \( K_S = \sum_{i=1}^{M} K_i \) can be written as:

\[
K_S(x, x') = \langle \Phi_S(x), \Phi_S(x') \rangle_{\mathcal{H}_S} ,
\]

where \( \Phi_S : X \rightarrow \mathcal{H}_S = \mathcal{H}_1 \oplus \ldots \oplus \mathcal{H}_M \) is the concatenation of the feature maps \( \Phi_i \):

\[
\Phi_S(x) = (\Phi_1(x), \ldots, \Phi_M(x))^\top .
\]

Therefore, summing kernels amounts to concatenating their feature space representations, which is a quite natural way to integrate different features.
Proof

For $\Phi_S(x) = (\Phi_1(x), \ldots, \Phi_M(x))^T$, we easily compute:

$$\langle \Phi_S(x), \Phi_S(x') \rangle_{H_s} = \sum_{i=1}^{M} \langle \Phi_i(x), \Phi_i(x') \rangle_{H_i}$$

$$= \sum_{i=1}^{M} K_i(x, x')$$

$$= K_S(x, x').$$
**Protein network inference from multiple genomic data: a supervised approach**

Y. Yamanishi¹,*, J.-P. Vert² and M. Kanehisa¹

¹Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611-0011, Japan and ²Computational Biology group, Ecole des Mines de Paris, 35 rue Saint-Honoré, 77305 Fontainebleau cedex, France

---

**Equation (2) with decreasing eigenvalue $\rho$.** This problem is usually called kernel canonical correlation analysis (CCA). To assess the similarity of protein networks, we simply follow the spirit of kernel PCA, and supervised approach based on kernel CCA.

**$K_{\text{exp}}$ (Expression)\n
$K_{\text{ppi}}$ (Protein interaction)\n
$K_{\text{loc}}$ (Localization)\n
$K_{\text{phy}}$ (Phylogenetic profile)\n
$K_{\text{exp}} + K_{\text{ppi}} + K_{\text{loc}} + K_{\text{phy}}$ (Integration)
The sum kernel: functional point of view

Theorem

The solution \( f^* \in \mathcal{H}_{K_S} \) when we learn with \( K_S = \sum_{i=1}^{M} K_i \) is equal to:

\[
f^* = \sum_{i=1}^{M} f_i^*,
\]

where \( (f_1^*, \ldots, f_M^*) \in \mathcal{H}_{K_1} \times \ldots \times \mathcal{H}_{K_M} \) is the solution of:

\[
\min_{f_1, \ldots, f_M} R \left( \sum_{i=1}^{M} f_i^n \right) + \lambda \sum_{i=1}^{M} \| f_i \|^2_{\mathcal{H}_{K_i}}.
\]
Generalization: The weighted sum kernel

**Theorem**

The solution $f^*$ when we learn with $K_\eta = \sum_{i=1}^{M} \eta_i K_i$, with $\eta_1, \ldots, \eta_M \geq 0$, is equal to:

$$f^* = \sum_{i=1}^{M} f_i^* ,$$

where $(f_1^*, \ldots, f_M^*) \in \mathcal{H}_{K_1} \times \ldots \times \mathcal{H}_{K_M}$ is the solution of:

$$\min_{f_1, \ldots, f_M} R \left( \sum_{i=1}^{M} f_i^n \right) + \lambda \sum_{i=1}^{M} \frac{\| f_i \|^2_{\mathcal{H}_{K_i}}}{\eta_i} .$$
Proof (1/4)

\[
\min_{f_1, \ldots, f_M} R \left( \sum_{i=1}^{M} f_i^n \right) + \lambda \sum_{i=1}^{M} \frac{\| f_i \|_{H_{K_i}}^2}{\eta_i}.
\]

- \( R \) being convex, the problem is strictly convex and has a unique solution \((f_1^*, \ldots, f_M^*) \in \mathcal{H}_{K_1} \times \ldots \times \mathcal{H}_{K_M}\).

- By the representer theorem, there exists \(\alpha_1^*, \ldots, \alpha_M^* \in \mathbb{R}^n\) such that

\[
\begin{align*}
  f_i^*(x) &= \sum_{j=1}^{n} \alpha_{ij}^* K_i(x_j, x).
\end{align*}
\]

- \((\alpha_1^*, \ldots, \alpha_M^*)\) is the solution of

\[
\min_{\alpha_1, \ldots, \alpha_M \in \mathbb{R}^n} R \left( \sum_{i=1}^{M} K_i \alpha_i \right) + \lambda \sum_{i=1}^{M} \frac{\alpha_i^\top K_i \alpha_i}{\eta_i}.
\]
Proof (2/4)

This is equivalent to

$$\min_{u, \alpha_1, \ldots, \alpha_M \in \mathbb{R}^n} R(u) + \lambda \sum_{i=1}^{M} \alpha_i^\top K_i \alpha_i \quad \text{s.t.} \quad u = \sum_{i=1}^{M} K_i \alpha_i .$$

This is equivalent to the saddle point problem:

$$\min_{u, \alpha_1, \ldots, \alpha_M \in \mathbb{R}^n} \max_{\gamma \in \mathbb{R}^n} R(u) + \lambda \sum_{i=1}^{M} \frac{\alpha_i^\top K_i \alpha_i}{\eta_i} + 2\lambda \gamma^\top (u - \sum_{i=1}^{M} K_i \alpha_i) .$$

By Slater’s condition, strong duality holds, meaning we can invert min and max:

$$\max_{\gamma \in \mathbb{R}^n} \min_{u, \alpha_1, \ldots, \alpha_M \in \mathbb{R}^n} R(u) + \lambda \sum_{i=1}^{M} \frac{\alpha_i^\top K_i \alpha_i}{\eta_i} + 2\lambda \gamma^\top (u - \sum_{i=1}^{M} K_i \alpha_i) .$$
Proof (3/4)

- Minimization in $u$:
  \[
  \min_u R(u) + 2\lambda\gamma^T u = -\max_u \left\{ -2\lambda\gamma^T u - R(u) \right\} = -R^*(-2\lambda\gamma),
  \]
  where $R^*$ is the Fenchel dual of $R$:

  \[
  \forall v \in \mathbb{R}^n \quad R^*(v) = \sup_{u \in \mathbb{R}^n} u^T v - R(u).
  \]

- Minimization in $\alpha_i$ for $i = 1, \ldots, M$:
  \[
  \min_{\alpha_i} \left\{ \lambda \frac{\alpha_i^T K_i \alpha_i}{\eta_i} - 2\lambda\gamma^T K_i \alpha_i \right\} = -\lambda \eta_i \gamma^T K_i \gamma,
  \]
  where the minimum in $\alpha_i$ is reached for $\alpha_i^* = \eta_i \gamma$. 

The dual problem is therefore
\[
\max_{\gamma \in \mathbb{R}^n} \left\{ -R^*(-2\lambda \gamma) - \lambda \gamma^\top \left( \sum_{i=1}^M \eta_i K_i \right) \gamma \right\}.
\]

Note that if learn from a single kernel $K_\eta$, we get the same dual problem
\[
\max_{\gamma \in \mathbb{R}^n} \left\{ -R^*(-2\lambda \gamma) - \lambda \gamma^\top K_\eta \gamma \right\}.
\]

If $\gamma^*$ is a solution of the dual problem, then $\alpha^*_i = \eta_i \gamma^*$ leading to:
\[
\forall x \in \mathcal{X}, \quad f^*_i (x) = \sum_{j=1}^n \alpha^*_i K_i (x_j, x) = \sum_{j=1}^n \eta_i \gamma^*_j K_i (x_j, x)
\]

Therefore, $f^* = \sum_{i=1}^M f^*_i$ satisfies
\[
f^* (x) = \sum_{i=1}^M \sum_{j=1}^n \eta_i \gamma^*_j K_i (x_j, x) = \sum_{j=1}^n \gamma^*_j K_\eta (x_j, x).
\]
Learning the kernel

Motivation

If we know how to weight each kernel, then we can learn with the weighted kernel

$$K_\eta = \sum_{i=1}^{M} \eta_i K_i$$

However, usually we don’t know...

Perhaps we can optimize the weights $\eta_i$ during learning?
An objective function for $K$

**Theorem**

For any p.d. kernel $K$ on $\mathcal{X}$, let

$$J(K) = \min_{f \in \mathcal{H}} \left\{ R(f^n) + \lambda \| f \|_{\mathcal{H}}^2 \right\}.$$  

The function $K \mapsto J(K)$ is convex.

This suggests a principled way to "learn" a kernel: define a convex set of candidate kernels, and minimize $J(K)$ by convex optimization.
Proof

- We have shown by strong duality that

\[ J(K) = \max_{\gamma \in \mathbb{R}^n} \left\{ -R^*(-2\lambda \gamma) - \lambda \gamma^T K \gamma \right\} . \]

- For each \( \gamma \) fixed, this is an affine function of \( K \), hence convex

A supremum of convex functions is convex. \( \square \)
MKL (Lanckriet et al., 2004)

- We consider the set of convex combinations

\[ K_\eta = \sum_{i=1}^{M} \eta_i K_i \quad \text{with} \quad \eta \in \Sigma_M = \left\{ \eta_i \geq 0, \sum_{i=1}^{M} \eta_i = 1 \right\} \]

- We optimize both \( \eta \) and \( f^* \) by solving:

\[
\min_{\eta \in \Sigma_M} J(K_\eta) = \min_{\eta \in \Sigma_M} \min_{f \in \mathcal{H}_{K_\eta}} \left\{ R(f^n) + \lambda \| f \|_{\mathcal{H}_{K_\eta}}^2 \right\}
\]

- The problem is jointly convex in \((\eta, \alpha)\) and can be solved efficiently.
- The output is both a set of weights \( \eta \), and a predictor corresponding to the kernel method trained with kernel \( K_\eta \).
- This method is usually called Multiple Kernel Learning (MKL).
A statistical framework for genomic data fusion

Gert R. G. Lanckriet¹, Tijl De Bie², Nello Cristianini³, Michael I. Jordan² and William Stafford Noble⁵,*

¹Department of Electrical Engineering and Computer Science, ²Division of Computer Science, Department of Statistics, University of California, Berkeley 94720, USA, ³Department of Electrical Engineering, ESAT-SCD, Katholieke Universiteit Leuven 3001, Belgium, ⁴Department of Statistics, University of California, Davis 95618, USA and ⁵Department of Genome Sciences, University of Washington, Seattle 98195, USA

Example: protein annotation

Table 1. Kernel functions

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Data</th>
<th>Similarity measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>K_SW</td>
<td>protein sequences</td>
<td>Smith-Waterman</td>
</tr>
<tr>
<td>K_B</td>
<td>protein sequences</td>
<td>BLAST</td>
</tr>
<tr>
<td>K_Pfam</td>
<td>protein sequences</td>
<td>Pfam HMM</td>
</tr>
<tr>
<td>K_FFT</td>
<td>hydropathy profile</td>
<td>FFT</td>
</tr>
<tr>
<td>K_LI</td>
<td>protein interactions</td>
<td>linear kernel</td>
</tr>
<tr>
<td>K_D</td>
<td>protein interactions</td>
<td>diffusion kernel</td>
</tr>
<tr>
<td>K_E</td>
<td>gene expression</td>
<td>radial basis kernel</td>
</tr>
<tr>
<td>K_RND</td>
<td>random numbers</td>
<td>linear kernel</td>
</tr>
</tbody>
</table>

Fig. 1. Overall quality of the ranking induced by the classifier, rather than looking at individual fold changes, allows for an unbiased analysis of the overall quality of the ranking. We define three kernel matrices based upon standard homology, BLAST, Pfam hmm, and diffusion kernel. The kernel function thus extracts a specific type of information from the data as strings from a 20-symbol alphabet; gene sequences are given in the online supplement.

The recent availability of multiple types of genome-wide data for integrating and drawing inferences from a collection of types of experimental data.

The approach relies on the use of kernel-based interactions to be informative for two reasons. First, hydrophobicity profiles are pre-filtered with a low-pass filter to reduce noise: generated from the Kyte–Doolittle index (Kyte and Doolittle, 1982). This kernel compares the frequency content of the hydropathy profiles (and if necessary appending zeros to make them equal in length—a commonly used technique not necessary for our experiments).

The width of those peaks above a certain threshold (Chen et al, 2000) leads to valid kernel matrices, computationally known the protein it encodes, that protein’s similarity to a row or column from the interaction matrix) for a pair of proteins, the larger the inner product will be.

Example: protein annotation: of proteins, ranked by the number of predicted transmembrane helices, the number of transmembrane helices predicted by TMHMM output. Accordingly, we produce two lists of transmembrane helices, the second one being the list of transmembrane helices predicted by TMHMM output. TMHMM output is the list of transmembrane helices predicted by TMHMM output.

The filter is intended to smooth out periodic features in the data, such as peaks and valleys. The filter is applied by convolving the data with a suitable kernel function, which is a weighted average of the data points around each point. The weights are determined by the kernel function, which is a function of the distance between the point and its neighbors. The weights are typically chosen to be smaller for points further away from the point of interest, so that the effect of those points is smaller.

The kernel functions derived from different types of data can be combined into a single kernel function, which is a linear combination of the individual kernel functions. The weights of the individual kernel functions are determined by solving an optimization problem, which is a convex optimization problem. The optimization problem is solved using semidefinite programming, which is a mathematical optimization technique.

The weights of the individual kernel functions are determined by solving an optimization problem, which is a convex optimization problem. The optimization problem is solved using semidefinite programming, which is a mathematical optimization technique.

The computational experiments performed using yeast genome-wide measurements. Each dataset is represented via a kernel function, which defines a similarity relation between pairs of entities, such as genes or proteins. The kernel function takes relationships that are implicit in the data and formalizes them into a similarity measure, which is a function that assigns a similarity score to each pair of entities.

The statistical framework for genomic data fusion that is presented in this paper is based on the use of kernel methods, which are a class of algorithms for pattern recognition, where the input data is mapped into a high-dimensional feature space through a nonlinear function, called the kernel function. The kernel function is a similarity measure that is defined in the feature space.

The similarities can be quite complex relations, implicitly captured and high-confidence interactions from a database of known interactions make them equal in length—a commonly used technique not necessary for our experiments.

The comparison of the performance of the different views of the genome that are provided by various types of genome-wide data is a challenging task, as the different views of the genome are not necessarily consistent with each other. A statistical framework for genomic data fusion that is presented in this paper is based on the use of kernel methods, which are a class of algorithms for pattern recognition, where the input data is mapped into a high-dimensional feature space through a nonlinear function, called the kernel function. The kernel function is a similarity measure that is defined in the feature space.

The comparison of the performance of the different views of the genome that are provided by various types of genome-wide data is a challenging task, as the different views of the genome are not necessarily consistent with each other. A statistical framework for genomic data fusion that is presented in this paper is based on the use of kernel methods, which are a class of algorithms for pattern recognition, where the input data is mapped into a high-dimensional feature space through a nonlinear function, called the kernel function. The kernel function is a similarity measure that is defined in the feature space.

The comparison of the performance of the different views of the genome that are provided by various types of genome-wide data is a challenging task, as the different views of the genome are not necessarily consistent with each other. A statistical framework for genomic data fusion that is presented in this paper is based on the use of kernel methods, which are a class of algorithms for pattern recognition, where the input data is mapped into a high-dimensional feature space through a nonlinear function, called the kernel function. The kernel function is a similarity measure that is defined in the feature space.

The comparison of the performance of the different views of the genome that are provided by various types of genome-wide data is a challenging task, as the different views of the genome are not necessarily consistent with each other. A statistical framework for genomic data fusion that is presented in this paper is based on the use of kernel methods, which are a class of algorithms for pattern recognition, where the input data is mapped into a high-dimensional feature space through a nonlinear function, called the kernel function. The kernel function is a similarity measure that is defined in the feature space.

The comparison of the performance of the different views of the genome that are provided by various types of genome-wide data is a challenging task, as the different views of the genome are not necessarily consistent with each other. A statistical framework for genomic data fusion that is presented in this paper is based on the use of kernel methods, which are a class of algorithms for pattern recognition, where the input data is mapped into a high-dimensional feature space through a nonlinear function, called the kernel function. The kernel function is a similarity measure that is defined in the feature space.
Example: 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 by MKL (M).

![Test error vs Kernels](chart.png)

Performance comparison on Corel14
MKL revisited (Bach et al., 2004)

\[ K_\eta = \sum_{i=1}^{M} \eta_i K_i \quad \text{with} \quad \eta \in \Sigma_M = \left\{ \eta_i \geq 0, \sum_{i=1}^{M} \eta_i = 1 \right\} \]

**Theorem**

The solution \( f^* \) of

\[
\min_{\eta \in \Sigma_M} \min_{f \in \mathcal{H}_{K_\eta}} \left\{ R(f^n) + \lambda \| f \|_{\mathcal{H}_{K_\eta}}^2 \right\}
\]

is \( f^* = \sum_{i=1}^{M} f_i^* \), where \((f_1^*, \ldots, f_M^*) \in \mathcal{H}_{K_1} \times \ldots \times \mathcal{H}_{K_M}\) is the solution of:

\[
\min_{f_1, \ldots, f_M} \left\{ R \left( \sum_{i=1}^{M} f_i^n \right) + \lambda \left( \sum_{i=1}^{M} \| f_i \|_{\mathcal{H}_{K_i}} \right)^2 \right\}.
\]
Proof (1/2)

\[
\min_{\eta \in \Sigma_M} \min_{f \in \mathcal{H}_{K\eta}} \left\{ R(f^n) + \lambda \| f \|^2_{\mathcal{H}_{K\eta}} \right\}
\]

\[
= \min_{\eta \in \Sigma_M} \min_{f_1, \ldots, f_M} \left\{ R \left( \sum_{i=1}^M f_i^n \right) + \lambda \sum_{i=1}^M \frac{\| f_i \|^2_{\mathcal{H}_{K_i}}}{\eta_i} \right\}
\]

\[
= \min_{f_1, \ldots, f_M} \left\{ R \left( \sum_{i=1}^M f_i^n \right) + \lambda \min_{\eta \in \Sigma_M} \left\{ \sum_{i=1}^M \frac{\| f_i \|^2_{\mathcal{H}_{K_i}}}{\eta_i} \right\} \right\}
\]

\[
= \min_{f_1, \ldots, f_M} \left\{ R \left( \sum_{i=1}^M f_i^n \right) + \lambda \left( \sum_{i=1}^M \| f_i \|_{\mathcal{H}_{K_i}} \right)^2 \right\},
\]
Proof (2/2)

where the last equality results from:

\[ \forall a \in \mathbb{R}^M_+, \quad \left( \sum_{i=1}^M a_i \right)^2 = \inf_{\eta \in \Sigma_M} \sum_{i=1}^M \frac{a_i^2}{\eta_i}, \]

which is a direct consequence of the Cauchy-Schwarz inequality:

\[ \sum_{i=1}^M a_i = \sum_{i=1}^M \frac{a_i}{\sqrt{\eta_i}} \times \sqrt{\eta_i} \leq \left( \sum_{i=1}^M \frac{a_i^2}{\eta_i} \right)^{\frac{1}{2}} \left( \sum_{i=1}^M \eta_i \right)^{\frac{1}{2}}. \]
Algorithm: simpleMKL (Rakotomamonjy et al., 2008)

- We want to minimize in $\eta \in \Sigma_M$:

$$\min_{\eta \in \Sigma_M} J(K_\eta) = \min_{\eta \in \Sigma_M} \max_{\gamma \in \mathbb{R}^n} \left\{ -R^*(-2\lambda \gamma) - \lambda \gamma^\top K_\eta \gamma \right\}.$$

- For a fixed $\eta \in \Sigma_M$, we can compute $f(\eta) = J(K_\eta)$ by using a standard solver for a single kernel to find $\gamma^*$:

$$J(K_\eta) = -R^*(-2\lambda \gamma^*) - \lambda \gamma^*\top K_\eta \gamma^*.$$

- From $\gamma^*$ we can also compute the gradient of $J(K_\eta)$ with respect to $\eta$:

$$\frac{\partial J(K_\eta)}{\partial \eta_i} = -\lambda \gamma^*\top K_i \gamma^*.$$

- $J(K_\eta)$ can then be minimized on $\Sigma_M$ by a projected gradient or reduced gradient algorithm.
Sum kernel vs MKL

- Learning with the sum kernel (uniform combination) solves

\[
\min_{f_1, \ldots, f_M} \left\{ R \left( \sum_{i=1}^{M} f_i^n \right) + \lambda \sum_{i=1}^{M} \| f_i \|_{H_{K_i}}^2 \right\}.
\]

- Learning with MKL (best convex combination) solves

\[
\min_{f_1, \ldots, f_M} \left\{ R \left( \sum_{i=1}^{M} f_i^n \right) + \lambda \left( \sum_{i=1}^{M} \| f_i \|_{H_{K_i}} \right)^2 \right\}.
\]

- Although MKL can be thought of as optimizing a convex combination of kernels, it is more correct to think of it as a penalized risk minimization estimator with the \textit{group lasso} penalty:

\[
\Omega(f) = \min_{f_1 + \ldots + f_M = f} \sum_{i=1}^{M} \| f_i \|_{H_{K_i}}.
\]
Example: ridge vs LASSO regression

- Take $\mathcal{X} = \mathbb{R}^d$, and for $x = (x_1, \ldots, x_d)^\top$ consider the rank-1 kernels:
  \[ \forall i = 1, \ldots, d, \quad K_i (x, x') = x_i x'_i. \]
- A function $f_i \in \mathcal{H}_{K_i}$ has the form $f_i (x) = \beta_i x_i$, with $\| f_i \|_{\mathcal{H}_{K_i}} = |\beta_i|$
- The sum kernel is $K_S (x, x') = \sum_{i=1}^d x_i x'_i = x^\top x$, a function $\mathcal{H}_{K_S}$ is of the form $f (x) = \beta^\top x$, with norm $\| f \|_{\mathcal{H}_{K_S}} = \| \beta \|_{\mathbb{R}^d}$.
- Learning with the sum kernel solves a ridge regression problem:
  \[
  \min_{\beta \in \mathbb{R}^d} \left\{ R(X\beta) + \lambda \sum_{i=1}^d \beta_i^2 \right\}.
  \]
- Learning with MKL solves a LASSO regression problem:
  \[
  \min_{\beta \in \mathbb{R}^d} \left\{ R(X\beta) + \lambda \left( \sum_{i=1}^d |\beta_i| \right)^2 \right\}.
  \]
Extensions (Micchelli et al., 2005)

For $r > 0$, \( K_\eta = \sum_{i=1}^{M} \eta_i K_i \) with \( \eta \in \Sigma_M^r = \left\{ \eta_i \geq 0, \sum_{i=1}^{M} \eta_i^r = 1 \right\} \)

**Theorem**

The solution \( f^* \) of

\[
\min_{\eta \in \Sigma_M^r} \min_{f \in H_{K_\eta}} \left\{ R(f^n) + \lambda \| f \|_{H_{K_\eta}}^2 \right\}
\]

is \( f^* = \sum_{i=1}^{M} f_i^* \), where \((f_1^*, \ldots, f_M^*) \in H_{K_1} \times \ldots \times H_{K_M}\) is the solution of:

\[
\min_{f_1, \ldots, f_M} \left\{ R \left( \sum_{i=1}^{M} f_i^n \right) + \lambda \left( \sum_{i=1}^{M} \| f_i \|_{H_{K_i}}^{2r} \right)^{\frac{r+1}{r}} \right\}.
\]
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
5. The Kernel Jungle
6. Open Problems and Research Topics
   - Multiple Kernel Learning (MKL)
   - Large-scale learning with kernels
   - Foundations of deep learning from a kernel point of view
Open Problems and Research Topics

- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
  - Motivation
    - Interlude: Large-scale learning with linear models
    - Nyström approximations
    - Random Fourier features
- Foundations of deep learning from a kernel point of view
Motivation

Main problem
All methods we have seen require computing the $n \times n$ Gram matrix, which is infeasible when $n$ is significantly greater than 100,000 both in terms of memory and computation.

Solutions
- low-rank approximation of the kernel;
- random Fourier features.

The goal is to find an approximate embedding $\psi : \mathcal{X} \to \mathbb{R}^d$ such that

$$K(x, x') \approx \langle \psi(x), \psi(x') \rangle_{\mathbb{R}^d}.$$

and use large-scale optimization techniques dedicated to linear models!
Motivation

Then, functions $f$ in $\mathcal{H}$ may be approximated by linear ones in $\mathbb{R}^d$, e.g.,

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) \approx \left\langle \sum_{i=1}^{n} \alpha_i \psi(x_i), \psi(x) \right\rangle_{\mathbb{R}^d} = \langle w, \psi(x) \rangle_{\mathbb{R}^d}.$$

Then, the ERM problem

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

becomes, approximately,

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} L(y_i, w^\top \psi(x_i)) + \lambda \|w\|_2^2,$$

which we know how to solve when $n$ is large.
Outline

Open Problems and Research Topics

- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
  - Motivation
  - Interlude: Large-scale learning with linear models
    - Nyström approximations
    - Random Fourier features
- Foundations of deep learning from a kernel point of view
Interlude: Large-scale learning with linear models

Let us study for a while optimization techniques for minimizing large sums of functions

\[
\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{w}).
\]

Good candidates are

- stochastic optimization techniques;
- randomized incremental optimization techniques;

We will see a couple of such algorithms with their convergence rates and start with the (batch) gradient descent method.
Introduction of a few optimization principles

Why do we care about convexity?
Introduction of a few optimization principles

Why do we care about convexity?

Local observations give information about the global optimum

\[ \nabla f(w) = 0 \] is a necessary and sufficient optimality condition for differentiable convex functions;

it is often easy to upper-bound \( f(w) - f^* \).
Introduction of a few optimization principles

An important inequality for smooth convex functions

If $f$ is convex

\[ f(w) \geq f(w^0) + \nabla f(w^0)^\top (w - w^0); \]

linear approximation

this is an equivalent definition of convexity for smooth functions.
Introduction of a few optimization principles

An important inequality for smooth functions

If $\nabla f$ is $L$-Lipschitz continuous ($f$ does not need to be convex)

$$f(w) \leq g(w) = f(w^0) + \nabla f(w^0)^\top (w - w^0) + \frac{L}{2} \|w - w^0\|^2;$$

$$g(w) = C_{w^0} + \frac{L}{2} \|w^0 - (1/L)\nabla f(w^0) - w\|^2.$$
Introduction of a few optimization principles

An important inequality for smooth functions

If $\nabla f$ is $L$-Lipschitz continuous ($f$ does not need to be convex)

1. $f(w) \leq g(w) = f(w^0) + \nabla f(w^0)^\top (w - w^0) + \frac{L}{2} ||w - w^0||_2^2$;
2. $w^1 = w^0 - \frac{1}{L} \nabla f(w^0)$ (gradient descent step).
Introduction of a few optimization principles

Gradient Descent Algorithm

Assume that $f$ is convex and differentiable, and that $\nabla f$ is $L$-Lipschitz.

**Theorem**

Consider the algorithm

$$w^t \leftarrow w^{t-1} - \frac{1}{L} \nabla f(w^{t-1}).$$

Then,

$$f(w^t) - f^* \leq \frac{L \|w^0 - w^*\|^2}{2t}.$$

**Remarks**

- the convergence rate improves under additional assumptions on $f$ (strong convexity);
- some variants have a $O(1/t^2)$ convergence rate (Nesterov, 2004).
Proof (1/2)

Proof of the main inequality for smooth functions

We want to show that for all \( w \) and \( z \),

\[
f(w) \leq f(z) + \nabla f(z)^\top (w - z) + \frac{L}{2} ||w - z||_2^2.
\]
Proof (1/2)

Proof of the main inequality for smooth functions

We want to show that for all $w$ and $z$,

$$f(w) \leq f(z) + \nabla f(z)^\top (w - z) + \frac{L}{2} \|w - z\|_2^2.$$

By using Taylor’s theorem with integral form,

$$f(w) - f(z) = \int_0^1 \nabla f(tw + (1 - t)z)^\top (w - z) dt.$$ 

Then,

$$f(w) - f(z) - \nabla f(z)^\top (w - z) \leq \int_0^1 (\nabla f(tw + (1 - t)z) - \nabla f(z)) \nabla f(z)^\top (w - z) dt$$

$$\leq \int_0^1 |(\nabla f(tw + (1 - t)z) - \nabla f(z)) \nabla f(z)^\top (w - z)| dt$$

$$\leq \int_0^1 \|\nabla f(tw + (1 - t)z) - \nabla f(z)\|_2 \|w - z\|_2 dt \quad (C.-S.)$$

$$\leq \int_0^1 Lt \|w - z\|_2^2 dt = \frac{L}{2} \|w - z\|_2^2.$$
We have shown that for all \( w \),

\[
f(w) \leq g_t(w) = f(w^{t-1}) + \nabla f(w^{t-1})^\top (w - w^{t-1}) + \frac{L}{2} \|w - w^{t-1}\|_2^2.
\]

\( g_t \) is minimized by \( w^t \); it can be rewritten \( g_t(w) = g_t(w^t) + \frac{L}{2} \|w - w^t\|_2^2 \). Then,

\[
f(w^t) \leq g_t(w^t) = g_t(w^*) - \frac{L}{2} \|w^* - w^t\|_2^2
\]

\[
= f(w^{t-1}) + \nabla f(w^{t-1})^\top (w^* - w^{t-1}) + \frac{L}{2} \|w^* - w^{t-1}\|_2^2 - \frac{L}{2} \|w^* - w^t\|_2^2
\]

\[
\leq f^* + \frac{L}{2} \|w^* - w^{t-1}\|_2^2 - \frac{L}{2} \|w^* - w^t\|_2^2.
\]

By summing from \( t = 1 \) to \( T \), we have a telescopic sum

\[
T(f(w^T) - f^*) \leq \sum_{t=1}^{T} f(w^t) - f^* \leq \frac{L}{2} \|w^* - w^0\|_2^2 - \frac{L}{2} \|w^* - w^T\|_2^2.
\]
Introduction of a few optimization principles

An important inequality for smooth and $\mu$-strongly convex functions

If $\nabla f$ is $L$-Lipschitz continuous and $f$ $\mu$-strongly convex

\[ f(w) \leq f(w^0) + \nabla f(w^0)^\top (w - w^0) + \frac{L}{2} \| w - w^0 \|^2; \]
\[ f(w) \geq f(w^0) + \nabla f(w^0)^\top (w - w^0) + \frac{\mu}{2} \| w - w^0 \|^2; \]
Proposition

When $f$ is $\mu$-strongly convex, differentiable and $\nabla f$ is $L$-Lipschitz, the gradient descent algorithm with step-size $1/L$ produces iterates such that

$$f(w^t) - f^* \leq \left(1 - \frac{\mu}{L}\right)^t \frac{L\|w^0 - w^*\|_2^2}{2}.$$  

We call that a linear convergence rate.
Proof

We start from an inequality from the previous proof

\[ f(w^t) \leq f(w^{t-1}) + \nabla f(w^{t-1})^\top (w^* - w^{t-1}) + \frac{L}{2} \|w^* - w^{t-1}\|_2^2 - \frac{L}{2} \|w^* - w^t\|_2^2 \]

\[ \leq f^* + \frac{L - \mu}{2} \|w^* - w^{t-1}\|_2^2 - \frac{L}{2} \|w^* - w^t\|_2^2. \]

In addition, we have that \( f(w^t) \geq f^* + \frac{\mu}{2} \|w^t - w^*\|_2^2 \), and thus

\[ \|w^* - w^t\|_2^2 \leq \frac{L - \mu}{L + \mu} \|w^* - w^{t-1}\|_2^2 \]

\[ \leq \left( 1 - \frac{\mu}{L} \right) \|w^* - w^{t-1}\|_2^2. \]

Finally,

\[ f(w^t) - f^* \leq \frac{L}{2} \|w^t - w^*\|_2^2 \]

\[ \leq \left( 1 - \frac{\mu}{L} \right)^t \frac{L \|w^* - w^0\|_2^2}{2} \]
The stochastic (sub)gradient descent algorithm

Consider now the minimization of an expectation

\[
\min_{w \in \mathbb{R}^p} f(w) = \mathbb{E}_x[\ell(x, w)],
\]

To simplify, we assume that for all \( x \), \( w \mapsto \ell(x, w) \) is differentiable, but everything here is true for nonsmooth functions.

**Algorithm**

At iteration \( t \),

- Randomly draw one example \( x_t \) from the training set;
- Update the current iterate

\[
w^t \leftarrow w^{t-1} - \eta_t \nabla_w \ell(x_t, w_{t-1}).
\]

- Perform online averaging of the iterates (optional)

\[
\tilde{w}^t \leftarrow (1 - \gamma_t)\tilde{w}^{t-1} + \gamma_t w^t.
\]
The stochastic (sub)gradient descent algorithm

There are various learning rates strategies (constant, varying step-sizes), and averaging strategies. Depending on the problem assumptions and choice of $\eta_t$, $\gamma_t$, classical convergence rates may be obtained:

- $f(\tilde{w}^t) - f^* = O(1/\sqrt{t})$ for convex problems;
- $f(\tilde{w}^t) - f^* = O(1/t)$ for strongly-convex ones;

Remarks

- The convergence rates are not that great, but the complexity per-iteration is small (1 gradient evaluation for minimizing an empirical risk versus $n$ for the batch algorithm).
- When the amount of data is infinite, the method minimizes the expected risk.
- Choosing a good learning rate automatically is an open problem.
Randomized incremental algorithms (1/2)

Consider now the minimization of a large finite sum of smooth convex functions:

$$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(w),$$

A class of algorithms with low per-iteration complexity have been recently introduced that enjoy exponential (aka, linear) convergence rates for strongly-convex problems, e.g., SAG (Schmidt et al., 2016).

**SAG algorithm**

$$w^t \leftarrow w^{t-1} - \frac{\gamma}{L n} \sum_{i=1}^{n} y_i^t \quad \text{with} \quad y_i^t = \begin{cases} \nabla f_i(w^{t-1}) & \text{if } i = i_t \\ y_i^{t-1} & \text{otherwise} \end{cases}.$$  

See also SAGA (Defazio et al., 2014), SVRG (Xiao and Zhang, 2014), SDCA (Shalev-Shwartz and Zhang, 2015), MISO (Mairal, 2015);
Randomized incremental algorithms (2/2)

Many of these techniques are in fact performing SGD-types of steps

$$\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \eta_t \mathbf{g}_t,$$

where \( \mathbb{E}[\mathbf{g}_t|\mathbf{w}_{t-1}] = \nabla f(\mathbf{w}_{t-1}) \), but where the estimator of the gradient has lower variance than in SGD, see SVRG (Xiao and Zhang, 2014).

Typically, these methods have the convergence rate

$$f(\mathbf{w}_t) - f^* = O \left( \left( 1 - C \max \left( \frac{1}{n}, \frac{\mu}{L} \right) \right)^t \right)$$

Remarks

- their complexity per-iteration is independent of \( n \)!
- unlike SGD, they are often almost parameter-free.
- besides, they can be accelerated (Lin et al., 2015).
Conclusion

- we know how to deal with huge-scale linear problems;
- this is also useful to learn with kernels!
Open Problems and Research Topics

- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
  - Motivation
  - Interlude: Large-scale learning with linear models
  - Nyström approximations
  - Random Fourier features
- Foundations of deep learning from a kernel point of view
Nyström approximations: principle

Consider a p.d. kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and RKHS $\mathcal{H}$, with the mapping $\varphi : \mathcal{X} \to \mathcal{H}$ such that

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

The Nyström method consists of replacing any point $\varphi(x)$ in $\mathcal{H}$, for $x$ in $\mathcal{X}$ by its orthogonal projection onto a finite-dimensional subspace

$$\mathcal{F} := \text{Span} (f_1, \ldots, f_p) \quad \text{with } p \ll n,$$

where the $f_i$’s are anchor points in $\mathcal{H}$ (to be defined later).

Motivation

- This principle allows us to work explicitly in a finite-dimensional space; it was introduced several times in the kernel literature [Williams and Seeger, 2002], [Smola and Schölkopf, 2000], [Fine and Scheinberg, 2001].
Nyström approximations: principle

The orthogonal projection is defined as

\[ \Pi_{\mathcal{F}}[x] := \arg\min_{f \in \mathcal{F}} \| \varphi(x) - f \|_{\mathcal{H}}^2, \]
Nyström approximations: principle

The projection is equivalent to

\[ \Pi_{\mathcal{F}}[x] := \sum_{j=1}^{p} \beta_j^* f_j \quad \text{with} \quad \beta^* \in \arg\min_{\beta \in \mathbb{R}^p} \left\| \varphi(x) - \sum_{j=1}^{p} \beta_j f_j \right\|_H^2, \]

and \( \beta^* \) is the solution of the problem

\[ \min_{\beta \in \mathbb{R}^p} -2 \sum_{j=1}^{p} \beta_j \langle f_j, \varphi(x) \rangle_H + \sum_{j,l=1}^{p} \beta_j \beta_l \langle f_j, f_l \rangle_H, \]

or also

\[ \min_{\beta \in \mathbb{R}^p} -2 \sum_{j=1}^{p} \beta_j f_j(x) + \sum_{j,l=1}^{p} \beta_j \beta_l \langle f_j, f_l \rangle_H. \]
Nyström approximations: principle

Then, call \([K_f]_{jl} = \langle f_j, f_l \rangle_H\) and \(f(x) = [f_1(x), \ldots, f_p(x)]\) in \(\mathbb{R}^p\). The problem may be rewritten as

\[
\min_{\beta \in \mathbb{R}^p} -2\beta^\top f(x) + \beta^\top K_f \beta,
\]

and, assuming \(K_f\) to be non-singular for simplicity, the solution is \(\beta^*(x) = K_f^{-1} f(x)\). Then,

\[
\varphi(x) \approx \sum_{j=1}^{p} \beta^*_j(x)f_j,
\]

and

\[
\langle \varphi(x), \varphi(x') \rangle_H \approx \left\langle \sum_{j=1}^{p} \beta^*_j(x)f_j, \sum_{j=1}^{p} \beta^*_j(x')f_j \right\rangle_H
\]

\[
= \sum_{j,l=1}^{p} \beta^*_j(x)\beta^*_l(x')\langle f_j, f_l \rangle_H = \beta^*(x)^\top K_f \beta^*(x').
\]
Nyström approximations: principle

This allows us to define the mapping

$$\psi(x) = K_f^{1/2} \beta^*(x) = K_f^{-1/2} f(x),$$

and we have the approximation $K(x, x') \approx \langle \psi(x), \psi(x') \rangle_{\mathbb{R}^p}$.

Remarks

- the mapping provides low-rank approximations of the kernel matrix.

  Given an $n \times n$ Gram matrix $K$ computed on a training set $S = \{x_1, \ldots, x_n\}$, we have

  $$K \approx \psi(S)^\top \psi(S),$$

  where $\psi(S) := [\psi(x_1), \ldots, \psi(x_n)]$.

- the approximation has a geometric interpretation.

- We need to define a good strategy for choosing the $f_j$'s.
Nyström approximation via kernel PCA

Let us now try to learn the $f_j$’s given training data $x_1, \ldots, x_n$ in $X$:

$$\min_{f_1, \ldots, f_p \in \mathcal{H}} \sum_{i=1}^{n} \left\| \varphi(x_i) - \sum_{j=1}^{p} \beta_{ij} f_j \right\|^2_{\mathcal{H}}.$$

Using similar calculation as before, the objective is equivalent to

$$\min_{f_1, \ldots, f_p \in \mathcal{H}} \sum_{i=1}^{n} -2 \beta_i^\top f(x_i) + \beta_i^\top K_f \beta_i,$$

and, by minimizing with respect to all $\beta_i$ with $f$ fixed, we have that $\beta_i = K_f^{-1} f(x_i)$ (assuming $K_f$ to be invertible), which leads to

$$\max_{f_1, \ldots, f_p \in \mathcal{H}} \sum_{i=1}^{n} f(x_i)^\top K_f^{-1} f(x_i).$$
Nyström approximation via kernel PCA

Remember the objective:

$$\max_{f_1, \ldots, f_p \in \mathcal{H}} \sum_{i=1}^{n} f(x_i) \top K_f^{-1} f(x_i).$$

Consider an optimal solution $f^\star$ and compute the eigenvalue decomposition of $K_{f^\star} = U \Delta U^\top$. Then, define the functions

$$g^\star(x) := [g_1^\star(x), \ldots, g_p^\star(x)] = \Delta^{-1/2} U^\top f^\star(x).$$

The functions $g_j^\star$ are points in the RKHS $\mathcal{H}$ since they are linear combinations of the functions $f_j^\star$ in $\mathcal{H}$.
Nyström approximation via kernel PCA

Remember the objective:
\[
\max_{f_1,\ldots,f_p \in \mathcal{H}} \sum_{i=1}^{n} f(x_i)^\top K_f^{-1} f(x_i).
\]

Consider an optimal solution \( f^\star \) and compute the eigenvalue decomposition of \( K_{f^\star} = U \Delta U^\top \). Then, define the functions
\[
g^\star(x) := [g_1^\star(x), \ldots, g_p^\star(x)] = \Delta^{-1/2} U^\top f^\star(x).
\]

The functions \( g_j^\star \) are points in the RKHS \( \mathcal{H} \) since they are linear combinations of the functions \( f_j^\star \) in \( \mathcal{H} \).

Exercise: check that all we do here and in the next slides can be extended to deal with singular Gram matrices \( K_{f^\star} \) and \( K_f \).
Nyström approximation via kernel PCA

Besides, by construction

\[
[K_{g^*}]_{jl} := \langle g_j^*, g_l^* \rangle_H
\]

\[
= \left\langle \frac{1}{\sqrt{\Delta_{jj}}} \sum_{k=1}^{p} [U]_{kj} f_k^*, \frac{1}{\sqrt{\Delta_{ll}}} \sum_{k=1}^{p} [U]_{kl} f_k^* \right\rangle_H
\]

\[
= \frac{1}{\sqrt{\Delta_{jj}}} \frac{1}{\sqrt{\Delta_{ll}}} \sum_{k,k'=1}^{p} [U]_{kj} [U]_{k'l} \langle f_k^*, f_{k'}^* \rangle_H
\]

\[
= \frac{1}{\sqrt{\Delta_{jj}}} \frac{1}{\sqrt{\Delta_{ll}}} \sum_{k,k'=1}^{p} [U]_{kj} [U]_{k'l} [K_{f^*}]_{kk'}
\]

\[
= \frac{1}{\sqrt{\Delta_{jj}}} \frac{1}{\sqrt{\Delta_{ll}}} u_j^\top K_{f^*} u_l
\]

\[
= \delta_{j=l}.
\]
Nyström approximation via kernel PCA

Then, $K_{g^*} = I$ and $g^*$ is also a solution of the problem

$$\max_{f_1, \ldots, f_p \in \mathcal{H}} \sum_{i=1}^{n} f(x_i)^\top K_f^{-1} f(x_i),$$

since

$$f^*(x_i)^\top K_f^{-1} f^*(x_i) = f^*(x_i)^\top U \Delta^{-1} U^\top f^*(x_i)$$

$$= g^*(x_i)^\top g^*(x_i) = g^*(x_i)^\top K_{g^*} g^*(x_i),$$

and also a solution of the problem

$$\max_{g_1, \ldots, g_p \in \mathcal{H}} \sum_{j=1}^{p} \sum_{i=1}^{n} g_j(x_i)^2 \quad \text{s.t.} \quad g_j \perp g_k \quad \text{for} \quad k \neq j \quad \text{and} \quad \|g_j\|_{\mathcal{H}} = 1.$$
Nyström approximation via kernel PCA

Then, $K_{g^*} = I$ and $g^*$ is also a solution of the problem

$$\max_{f_1, \ldots, f_p \in \mathcal{H}} \sum_{i=1}^{n} f(x_i)^\top K_f^{-1} f(x_i),$$

since

$$f^*(x_i)^\top K_f^{-1} f^*(x_i) = f^*(x_i)^\top U \Delta^{-1} U^\top f^*(x_i) = g^*(x_i)^\top g^*(x_i) = g^*(x_i)^\top K_{g^*} g^*(x_i),$$

and also a solution of the problem

$$\max_{g_1, \ldots, g_p \in \mathcal{H}} \sum_{j=1}^{p} \sum_{i=1}^{n} g_j(x_i)^2 \quad \text{s.t.} \quad g_j \perp g_k \quad \text{for} \quad k \neq j \quad \text{and} \quad \|g_j\|_{\mathcal{H}} = 1.$$  

This is the kernel PCA formulation!
Our first recipe with kernel PCA

Given a dataset of $n$ training points $x_1, \ldots, x_n$ in $\mathcal{X}$,

- randomly choose a subset $Z = [x_{z1}, \ldots, x_{zm}]$ of $m \leq n$ training points;
- compute the $m \times m$ kernel matrix $K_Z$.
- perform kernel PCA to find the $p \leq m$ largest principal directions (parametrized by $p$ vectors $\alpha_j$ in $\mathbb{R}^m$);

Then, every point $x$ in $\mathcal{X}$ may be approximated by

$$
\psi(x) = K^{-1/2}_{g^*} g^*(x) = g^*(x) = [g_1^*(x), \ldots, g_p^*(x)]^\top
$$

$$
= \left[ \sum_{i=1}^{m} \alpha_{1i} K(x_{z_i}, x), \ldots, \sum_{i=1}^{m} \alpha_{pi} K(x_{z_i}, x) \right]^\top.
$$
Nyström approximation via kernel PCA

Remarks

- The vector $\psi(x)$ can be interpreted as coordinates of the projection of $\varphi(x)$ onto the (orthogonal) PCA basis.
- The complexity of training is $O(m^3)$ (eig decomposition of $K_Z$) + $O(m^2)$ kernel evaluations.
- The complexity of encoding a new point $x$ is $O(mp)$ (matrix vector multiplication) + $O(m)$ kernel evaluations.
Nyström approximation via kernel PCA

Remarks

- The vector $\psi(x)$ can be interpreted as coordinates of the projection of $\varphi(x)$ onto the (orthogonal) PCA basis.
- The complexity of training is $O(m^3)$ (eig decomposition of $K_Z$) + $O(m^2)$ kernel evaluations.
- The complexity of encoding a new point $x$ is $O(mp)$ (matrix vector multiplication) + $O(m)$ kernel evaluations.

The main issue is the encoding time, which depends linearly on $m > p$. 
Nyström approximation via random sampling

A popular alternative is instead to select the anchor points among the training data points \( x_1, \ldots, x_n \)—that is,

\[
\mathcal{F} := \text{span}(\varphi(x_{z_1}), \ldots, \varphi(z_{zp})).
\]

In other words, choose \( f_1 = \varphi(x_{z_1}), \ldots, f_p = \varphi(x_{zp}) \).

Second recipe with random point sampling

Given a dataset of \( n \) training points \( x_1, \ldots, x_n \) in \( \mathcal{X} \),

- randomly choose a subset \( \mathcal{Z} = [x_{z_1}, \ldots, x_{zp}] \) of \( p \) training points;
- compute the \( p \times p \) kernel matrix \( \mathbf{K}_{\mathcal{Z}} \).

Then, a new point \( x \) is encoded as

\[
\psi(x) = \mathbf{K}_{\mathcal{Z}}^{-1/2} \mathbf{f}_{\mathcal{Z}}(x) \\
= \mathbf{K}_{\mathcal{Z}}^{-1/2} [K(x_{z_1}, x), \ldots, K(x_{zp}, x)]^\top
\]
Nyström approximation via random sampling

- The complexity of training is $O(p^3)$ (eig decomposition) + $O(p^2)$ kernel evaluations.
- The complexity of encoding a point $x$ is $O(p^2)$ (matrix vector multiplication) + $O(p)$ kernel evaluations.
Nyström approximation via random sampling

- The complexity of training is $O(p^3)$ (eig decomposition) + $O(p^2)$ kernel evaluations.
- The complexity of encoding a point $x$ is $O(p^2)$ (matrix vector multiplication) + $O(p)$ kernel evaluations.

The main issue complexity is better, but we lose the “optimality” of the PCA basis and the random choice of anchor points is not clever.
Nyström approximation via greedy approach

Better approximation can be obtained with a greedy algorithm that iteratively selects one column at a time with largest residual (Bach and Jordan, 2002; Smola and Shölkopf, 2000, Fine and Scheinbert, 2000).

At iteration $k$, assume that $Z = \{x_{z1}, \ldots, x_{zk}\}$; then, the residual for a data point $x$ encoded with $k$ anchor points $f_1, \ldots, f_k$ is

$$\min_{\beta \in \mathbb{R}^k} \left\| \varphi(x) - \sum_{j=1}^{k} \beta_j \varphi(x_{zj}) \right\|_H^2,$$

which is equal to

$$\|\varphi(x)\|_H^2 - f_Z(x)^\top K_Z^{-1} f_Z(x),$$

and since $f_j = \varphi(x_{zj})$ for all $j$, the data point $x_i$ with largest residual is the one that maximizes

$$K(x_i, x_i) - f_Z(x_i)K_Z^{-1}f_Z(x_i) \quad \text{with} \quad f_Z(x_i) = [K(x_{z1}, x), \ldots, K(x_{zk}, x)]^\top.$$
Nyström approximation via greedy approach

This brings us to the following algorithm

**Third recipe with greedy anchor point selection**

Initialize $Z = \emptyset$. For $k = 1, \ldots, p$ do

- **data point selection**

  $$z_k \leftarrow \arg \max_{i \in \{1, \ldots, n\}} K(x_i, x_i) - f_Z(x_i)K_{Z}^{-1}f_Z(x_i);$$

- **update the set $Z$**

  $$Z \leftarrow Z \cup \{x_{z_k}\}.$$  

**Remarks**

- A naive implementation costs $O(k^2n + k^3)$ at every iteration.
- To get a reasonable complexity, one has to use simple linear algebra tricks (see next slide).
Nyström approximation via greedy approach

If $\mathcal{Z}' = \mathcal{Z} \cup \{z\}$,

$$K_{\mathcal{Z}'}^{-1} = \left[ \begin{array}{cc} K_{\mathcal{Z}} & f_{\mathcal{Z}}(z) \\ f_{\mathcal{Z}}(z)^{\top} & K(z, z) \end{array} \right]^{-1} = \left[ \begin{array}{cc} K_{\mathcal{Z}}^{-1} + \frac{1}{s}bb^{\top} & -\frac{1}{s}b \\ -\frac{1}{s}b^{\top} & \frac{1}{s} \end{array} \right],$$

where $s$ is the Schur complement $s = K(z, z) - f_{\mathcal{Z}}(z)K_{\mathcal{Z}}^{-1}f_{\mathcal{Z}}(z)$, and $b = K_{\mathcal{Z}}^{-1}f_{\mathcal{Z}}(z)$.

**Complexity analysis**

- $K_{\mathcal{Z}'}^{-1}$ can be obtained from $K_{\mathcal{Z}}^{-1}$ and $f_{\mathcal{Z}}(z)$ in $O(k^2)$ float operations; for that we need to always keep into memory the $n$ vectors $f_{\mathcal{Z}}(x_i)$.
- updating the $f_{\mathcal{Z}'}(x_i)$’s from $f_{\mathcal{Z}}(x_i)$ requires $n$ kernel evaluations;

The total training complexity is $O(p^2n)$ float operations and $O(pn)$ kernel evaluations.
Nyström approximation via K-means

When $\mathcal{X} = \mathbb{R}^d$, it is also possible to synthesize points $z_1, \ldots, z_p$ such that they represented well some training data $x_1, \ldots, x_n$, leading to the Clustred Nyström approximation (Zhang and Kwok, 2008).

### Fourth recipe with K-means

1. Perform the regular K-means algorithm on the training data, to obtain $p$ centroids $z_1, \ldots, z_p$ in $\mathbb{R}^p$.
2. Define the anchor points $f_j = \varphi(z_j)$ for $j = 1, \ldots, p$, and perform the classical Nyström approximation.

### Remarks

- The complexity is the same as Nyström with random selection (except for the K-means step);
- The method is data-dependent and can significantly outperform the other variants in practice.
Nyström approximation: conclusion

Concluding remarks

- The greedy selection rule is equivalent to computing an incomplete Cholesky factorization of the kernel matrix (Bach and Jordan, 2002; Scholköpf and Smola, 2000, Fine and Scheinberg, 2001);
- The techniques we have seen produce low-rank approximations of the kernel matrix $K \approx LL^T$;
- The method admits a geometric interpretation in terms of orthogonal projection onto a finite-dimensional subspace.
- The approximation provides points in the RKHS. As such, many operations on the mapping are valid (translations, linear combinations, projections), unlike the method that will come next.
Outline

6 Open Problems and Research Topics

- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
  - Motivation
  - Interlude: Large-scale learning with linear models
  - Nyström approximations
  - Random Fourier features
- Foundations of deep learning from a kernel point of view
A large class of approximations for shift-invariant kernels are based on sampling techniques. Consider a real-valued positive-definite continuous translation-invariant kernel $K(x, y) = \kappa(x - y)$ with $\kappa : \mathbb{R}^d \to \mathbb{R}$. Then, if $\kappa(0) = 1$, Bochner theorem tells us that $\kappa$ is a valid characteristic function for some probability measure

$$\kappa(z) = \mathbb{E}_w[e^{i w^\top z}].$$

Remember indeed that, with the right assumptions on $\kappa$,

$$\kappa(x - y) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{\kappa}(w) e^{i w^\top x} e^{-i w^\top y} dw,$$

and the probability measure admits a density $q(w) = \frac{1}{(2\pi)^d} \hat{\kappa}(w)$ (non-negative, real-valued, sum to 1 since $\kappa(0) = 1$).
Random Fourier features (2/5)

Then,

\[ \kappa(x - y) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{\kappa}(w) e^{iw^\top x} e^{-iw^\top y} dw \]

\[ \quad = \int_{\mathbb{R}^d} q(w) \cos(w^\top x - w^\top y) dw \]

\[ \quad = \int_{\mathbb{R}^d} q(w) \left( \cos(w^\top x) \cos(w^\top y) + \sin(w^\top x) \sin(w^\top y) \right) dw \]

\[ \quad = \int_{\mathbb{R}^d} \int_{b=0}^{2\pi} \frac{q(w)}{2\pi} 2 \cos(w^\top x + b) \cos(w^\top y + b) dw db \quad \text{(exercise)} \]

\[ \quad = \mathbb{E}_{w \sim q(w), b \sim \mathcal{U}[0, 2\pi]} \left[ \sqrt{2} \cos(w^\top x + b) \sqrt{2} \cos(w^\top y + b) \right] \]
Random Fourier features (3/5)

Random Fourier features recipe

- Compute the Fourier transform of the kernel \( \hat{\kappa} \) and define the probability density \( q(\mathbf{w}) = \hat{\kappa}(\mathbf{w})/(2\pi)^d \);
- Draw \( p \) i.i.d. samples \( \mathbf{w}_1, \ldots, \mathbf{w}_p \) from \( q \) and \( p \) i.i.d. samples \( b_1, \ldots, b_p \) from the uniform distribution on \([0, 2\pi]\);
- Define the mapping

\[
\mathbf{x} \mapsto \psi(\mathbf{x}) = \sqrt{\frac{2}{d}} \left[ \cos(\mathbf{w}_1^\top \mathbf{x} + b_1), \ldots, \cos(\mathbf{w}_p^\top \mathbf{x} + b_p) \right]^\top.
\]

Then, we have that

\[
\kappa(\mathbf{x} - \mathbf{y}) \approx \langle \psi(\mathbf{x}), \psi(\mathbf{y}) \rangle_{\mathbb{R}^p}.
\]

The two quantities are equal in expectation.
Random Fourier features (4/5)

Theorem, [Rahimi and Recht, 2007]

On any compact subset $\mathcal{X}$ of $\mathbb{R}^m$, for all $\varepsilon > 0$,

$$
\mathbb{P} \left[ \sup_{x, y \in \mathcal{X}} \left| \kappa(x - y) - \langle \psi(x), \psi(y) \rangle_{\mathbb{R}^p} \right| \geq \varepsilon \right] \leq 2^8 \left( \frac{\sigma_q \text{diam}(\mathcal{X})}{\varepsilon} \right)^2 e^{-\frac{p\varepsilon^2}{4(m+2)}},
$$

where $\sigma_q^2 = \mathbb{E}_{w \sim q(w)}[w^\top w]$ is the second moment of the Fourier transform of $\kappa$.

Remarks

- The convergence is uniform, not data dependent;
- Take the sequence $\varepsilon_p = \sqrt{\frac{\log(p)}{p}} \sigma_q \text{diam}(\mathcal{X})$; Then the term on the right converges to zero when $p$ grows to infinity;
- Prediction functions with Random Fourier features are not in $\mathcal{H}$. 
Random Fourier features (5/5)

Ingredients of the proof

- For a \textit{fixed} pair of points \(x, y\), Hoeffding’s inequality says that
  \[
  \mathbb{P}\left[ \left| \kappa(x - y) - \langle \psi(x), \psi(y) \rangle_{\mathbb{R}^d} \right| \geq \varepsilon \right] \leq 2e^{-\frac{p\varepsilon^2}{4}}.
  \]

- Consider a net (set of balls of radius \(r\)) that covers \(X' = \{x - y : (x, y) \in X\}\) with at most \(T = (4\text{diam}(X')/r)^m\) balls.

- Apply the Hoeffding’s inequality to the centers \(x_i - y_i\) of the balls;

- Use a basic union bound
  \[
  \mathbb{P}\left[ \sup_i f(x_i, y_i) \geq \frac{\varepsilon}{2} \right] \leq \sum_i \mathbb{P}\left[ f(x_i, y_i) \geq \frac{\varepsilon}{2} \right] \leq 2Te^{-\frac{p\varepsilon^2}{8}}.
  \]

- Glue things together: control the probability for points \((x, y)\) inside each ball, and adjust the radius \(r\) (a bit technical).
Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
5. The Kernel Jungle
6. Open Problems and Research Topics
   - Multiple Kernel Learning (MKL)
   - Large-scale learning with kernels
   - Foundations of deep learning from a kernel point of view
Open Problems and Research Topics

- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
- Foundations of deep learning from a kernel point of view
  - Motivation
  - Deep kernel machines
  - Deep learning and stability
  - Application to graphs
  - Application to biological sequences
Understanding deep learning

The challenge of deep learning theory

- **Over-parameterized** (millions of parameters)
- **Expressive** (can approximate any function)
- Complex architectures for exploiting problem structure
- Yet, **easy to optimize** with (stochastic) gradient descent!
Understanding deep learning

The challenge of deep learning theory

- **Over-parameterized** (millions of parameters)
- **Expressive** (can approximate any function)
- Complex **architectures** for exploiting problem structure
- Yet, **easy to optimize** with (stochastic) gradient descent!

A functional space viewpoint

- View deep networks as functions in some functional space;
- Non-parametric models, natural measures of complexity (e.g., norms).
Understanding deep learning

The challenge of deep learning theory

- **Over-parameterized** (millions of parameters)
- **Expressive** (can approximate any function)
- Complex architectures for exploiting problem structure
- Yet, **easy to optimize** with (stochastic) gradient descent!

A functional space viewpoint

- View deep networks as functions in some functional space;
- Non-parametric models, natural measures of complexity (e.g., norms).

What is an appropriate functional space?
Success of deep learning

What can I help you with?

where is the train station?  où est la gare?
In the context of supervised learning

The goal is to learn a **prediction function** \( f : \mathcal{X} \rightarrow \mathcal{Y} \) given labeled training data \((x_i, y_i)_{i=1,...,n}\) with \(x_i\) in \(\mathcal{X}\), and \(y_i\) in \(\mathcal{Y}\):

\[
\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \Omega(f).
\]

What is specific to multilayer neural networks?

- The “neural network” space \(\mathcal{F}\) is explicitly parametrized by:
  \[
  f(x) = \sigma_k(A_k\sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2\sigma_1(A_1x))) \ldots)).
  \]
- Linear operations are either unconstrained (fully connected) or involve parameter sharing (e.g., convolutions).
- Finding the optimal \(A_1, A_2, \ldots, A_k\) yields a **non-convex** optimization problem.
What are the main features of CNNs?

- they capture **compositional** and **multiscale** structures in images;
- they provide some **invariance**;
- they model the **local stationarity** of images at several scales;
What are the main features of CNNs?

- they capture *compositional* and *multiscale* structures in images;
- they provide some *invariance*;
- they model the *local stationarity* of images at several scales;
CNNs (Picture from unknown source)

ImageNet: 1000 image categories, 10M hand-labeled images; top-5 error rate.

Figure: Top-5 error rate
Convolutional neural networks for biological sequences

Figure: two-layer CNN architecture from Alipanahi et al. (2015)

- Sequences are represented by one-hot encoding $(A=(1,0,0,0), C=(0,1,0,0), \ldots)$. 
- Single convolution layer followed by linear classifier.
Convolutional Neural Networks

What are current important problems to solve?

1. lack of **stability and robustness** (see next slide).
2. learning without **large amounts of data**.
3. making **interpretable** decisions.
4. ...
Figure: Adversarial examples are generated by computer; then printed on paper; a new picture taken on a smartphone fools the classifier.
Adversarial examples

(b)

clean + noise $\rightarrow$ “ostrich” (Szegedy et al., 2013).
Adversarial examples

(a real ostrich)
Adversarial examples

88% tabby cat  adversarial perturbation  99% guacamole

https://github.com/anishathalye/obfuscated-gradients
Convolutional Neural Networks

\[ \min_{f \in \mathcal{F}} \left( \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \frac{\lambda}{\Omega} \right) \]

The issue of regularization

- today, heuristics are used (DropOut, weight decay, early stopping)...
- ...but they are not sufficient.
- how to control variations of prediction functions?

\[ |f(x) - f(x')| \] should be close if \( x \) and \( x' \) are “similar”.

- what does it mean for \( x \) and \( x' \) to be “similar”?
- what should be a good regularization function \( \Omega \)?
Open Problems and Research Topics
- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
- Foundations of deep learning from a kernel point of view
  - Motivation
  - Deep kernel machines
  - Deep learning and stability
  - Application to graphs
  - Application to biological sequences
Relevant concepts

- **Dot-product kernels**:
  \[
  K(x, x') = \kappa(x^\top x') \quad \text{or} \quad K(x, x') = \|x\|\|x'\|\kappa\left(\frac{x^\top x'}{\|x\|\|x'\|}\right)
  \]

- **Hierarchical composition of feature spaces**:
  \[
  K(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad \text{with} \quad \Phi(x) = \varphi_2(\varphi_1(x))
  \]

- **NTK**: Asymptotic behavior of over-parametrized deep neural networks learned by gradient descent.

- **CKN**: Convolutional and hierarchical kernel constructions + end-to-end learning with kernels.
Relevant concepts

- **Dot-product kernels:**
  \[ K(x, x') = \kappa(x^\top x') \quad \text{or} \quad K(x, x') = \|x\|\|x'\|\kappa \left( \frac{x^\top x'}{\|x\|\|x'\|} \right) \]

- **Hierarchical composition of feature spaces:**
  \[ K(x, x') = \langle \Phi(x), \Phi(x') \rangle \text{ with } \Phi(x) = \varphi_2(\varphi_1(x)) \]

- **NTK:** Asymptotic behavior of over-parametrized deep neural networks learned by gradient descent.

- **CKN:** Convolutional and hierarchical kernel constructions + end-to-end learning with kernels.

What does it mean to do end-to-end learning with kernels?
Hierarchical kernels (Cho and Saul, 2009b)

- Kernels can be constructed hierarchically

\[ K(x, x') = \langle \Phi(x), \Phi(x') \rangle \text{ with } \Phi(x) = \varphi_2(\varphi_1(x)) \]

- e.g., dot-product kernels on the sphere

\[ K(x, x') = \kappa_2(\langle \varphi_1(x), \varphi_1(x') \rangle) = \kappa_2(\kappa_1(x^\top x')) \]
Kernels for deep models: deep kernel machines

Hierarchical kernels (Cho and Saul, 2009b)

- Kernels can be constructed **hierarchically**

\[ K(x, x') = \langle \Phi(x), \Phi(x') \rangle \text{ with } \Phi(x) = \varphi_2(\varphi_1(x)) \]

- e.g., dot-product kernels on the sphere

\[ K(x, x') = \kappa_2(\langle \varphi_1(x), \varphi_1(x') \rangle) = \kappa_2(\kappa_1(x^\top x')) \]

A classical old result (Schoenberg, 1942)

Let \( \mathcal{X} = \mathbb{S} \) be the unit sphere of some Hilbert space \( \mathcal{H}_0 \). The kernel \( K : \mathcal{X}^2 \to \mathbb{R} \)

\[ K(x, y) = \kappa(\langle x, y \rangle_{\mathcal{H}_0}), \]

is positive definite for all \( \mathcal{H}_0 \) if and only if \( \kappa \) is smooth and admits an expansion \( \kappa(u) = \sum_i a_i u^i \) with non-negative coefficients \( a_i \).
## Kernels for deep models: dot-product kernels

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>linear kernel</strong></td>
<td>$\langle z, z' \rangle$</td>
</tr>
<tr>
<td><strong>exponential kernel</strong></td>
<td>$e^{\alpha(\langle z, z' \rangle - 1)}$</td>
</tr>
<tr>
<td><strong>inverse polynomial kernel</strong></td>
<td>$\frac{1}{2 - \langle z, z' \rangle}$</td>
</tr>
<tr>
<td><strong>polynomial kernel of degree $p$</strong></td>
<td>$(c + \langle z, z' \rangle)^p$</td>
</tr>
</tbody>
</table>
| **arc-cosine kernel of degree 1** | $\frac{1}{\pi} (\sin(\theta) + (\pi - \theta) \cos(\theta))$  
with $\theta = \arccos(\langle z, z' \rangle)$ |
| **Vovk’s kernel of degree 3** | $\frac{1}{3} \left( \frac{1 - \langle z, z' \rangle^3}{1 - \langle z, z' \rangle} \right) = \frac{1}{3} \left( 1 + \langle z, z' \rangle + \langle z, z' \rangle^2 \right)$ |
Kernels for deep models: dot-product kernels

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear kernel</td>
<td>$\langle z, z' \rangle$</td>
</tr>
<tr>
<td>exponential kernel</td>
<td>$e^{\alpha(\langle z, z' \rangle - 1)}$</td>
</tr>
<tr>
<td>inverse polynomial kernel</td>
<td>$\frac{1}{2 - \langle z, z' \rangle}$</td>
</tr>
<tr>
<td>polynomial kernel of degree $p$</td>
<td>$(c + \langle z, z' \rangle)^p$</td>
</tr>
<tr>
<td>arc-cosine kernel of degree 1</td>
<td>$\frac{1}{\pi} \left( \sin(\theta) + (\pi - \theta) \cos(\theta) \right)$ with $\theta = \arccos(\langle z, z' \rangle)$</td>
</tr>
<tr>
<td>Vovk’s kernel of degree 3</td>
<td>$\frac{1}{3} \left( \frac{1 - \langle z, z' \rangle^3}{1 - \langle z, z' \rangle} \right) = \frac{1}{3} \left( 1 + \langle z, z' \rangle + \langle z, z' \rangle^2 \right)$</td>
</tr>
</tbody>
</table>

**Remark**

If $\|z\| = \|z'\| = 1$, the exponential kernel recovers the Gaussian kernel

$$\kappa_{\text{exp}}(\langle z, z' \rangle) = e^{\alpha(\langle z, z' \rangle - 1)} = e^{-\frac{\alpha}{2} \|z - z'\|^2},$$
Kernels for deep models: random feature kernels

\[ f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \quad m \to \infty \]

**Random feature kernels** (RF, Neal, 1996; Rahimi and Recht, 2007)

- \( \theta = (v_i)_i \), fixed random weights \( w_i \sim N(0, I) \)

\[ K_{RF}(x, y) = \mathbb{E}_{w \sim N(0, I)}[\sigma(w^\top x)\sigma(w^\top y)] \]
Kernels for deep models: random feature kernels

\[ f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \quad m \to \infty \]

Random feature kernels (RF, Neal, 1996; Rahimi and Recht, 2007)

- \( \theta = (v_i)_i \), fixed random weights \( w_i \sim N(0, I) \)

\[ K_{RF}(x, y) = \mathbb{E}_{w \sim N(0, I)}[\sigma(w^\top x)\sigma(w^\top y)] \]

- integral representations are not only available for t.i. kernels. They also work for several dot-product kernels (Cho and Saul, 2009b):

\[ k_n(x, y) = \frac{1}{\pi} \|x\|^n \|y\|^n J_n(\theta) \quad \text{with} \quad \theta = \cos^{-1}\left(\frac{x^\top y}{\|x\| \|y\|}\right) \]

with

\[ J_n(\theta) = (-1)^n (\sin \theta)^{2n+1} \left(\frac{1}{\sin \theta} \frac{\partial}{\partial \theta}\right)^n \left(\frac{\pi - \theta}{\sin \theta}\right) \]
Kernels for deep models: random feature kernels

\[ f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \quad m \to \infty \]

**Random feature kernels** (RF, Neal, 1996; Rahimi and Recht, 2007)

- \( \theta = (v_i)_i \), fixed random weights \( w_i \sim N(0, I) \)

\[
K_{RF}(x, y) = \mathbb{E}_{w \sim N(0, I)}[\sigma(w^\top x)\sigma(w^\top y)]
\]

- Integral representations are not only available for t.i. kernels. They also work for several dot-product kernels (Cho and Saul, 2009b):

\[
k_n(x, y) = \frac{1}{\pi} \|x\|^n \|y\|^n J_n(\theta) \quad \text{with} \quad \theta = \cos^{-1}\left(\frac{x^\top y}{\|x\|\|y\|}\right)
\]

with

\[
\begin{align*}
J_0(\theta) &= \pi - \theta \\
J_1(\theta) &= \sin(\theta) + (\pi - \theta) \cos(\theta) \\
J_2(\theta) &= 3 \sin(\theta) \cos(\theta) + (\pi - \theta)(1 + 2 \cos^2(\theta))
\end{align*}
\]
Kernels for deep models: random feature kernels

Theorem, (Cho and Saul, 2009a)

Consider

\[ k_n(x, y) = \frac{1}{\pi} \|x\|^n \|y\|^n J_n(\theta) \quad \text{with} \quad \theta = \cos^{-1} \left( \frac{x^\top y}{\|x\| \|y\|} \right). \]

Then

\[ k_n(x, y) = \mathbb{E}_{w \sim \mathcal{N}(0, I)}[\sigma(w^\top x)\sigma(w^\top y)], \]

with \( \sigma(u) = \frac{u^n}{\sqrt{2}}(1 + \text{sign}(u)) \).

- Note that \( k_1(x, y) = \mathbb{E}_{w \sim \mathcal{N}(0, I)}[\text{RELU}(w^\top x)\text{RELU}(w^\top y)] \).
- One of the fundamental tool to analyze RELU networks.
Kernels for deep models: neural tangent kernels

\[ f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \quad m \to \infty \]

**Neural tangent kernels** (NTK, Jacot et al., 2018)

- \( \theta = (v_i, w_i)_i \), initialization \( \theta_0 \sim N(0, I) \)
- **Lazy training** (Chizat et al., 2019): \( \theta \) stays close to \( \theta_0 \) when training with large \( m \)

\[ f_\theta(x) \approx f_{\theta_0}(x) + \langle \theta - \theta_0, \nabla_\theta f_\theta(x) |_{\theta=\theta_0} \rangle. \]

- Gradient descent for \( m \to \infty \approx \text{kernel ridge regression with neural tangent kernel} \)
Kernels for deep models: neural tangent kernels

\[ f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \quad m \to \infty \]

Neural tangent kernels (NTK, Jacot et al., 2018)
- \( \theta = (v_i, w_i)_i \), initialization \( \theta_0 \sim N(0, I) \)
- Lazy training (Chizat et al., 2019): \( \theta \) stays close to \( \theta_0 \) when training with large \( m \)

\[ f_\theta(x) \approx f_{\theta_0}(x) + \langle \theta - \theta_0, \nabla_\theta f_\theta(x) |_{\theta=\theta_0} \rangle. \]

- Gradient descent for \( m \to \infty \approx \) kernel ridge regression with neural tangent kernel

\[ K_{NTK}(x, y) = \lim_{m \to \infty} \langle \nabla_\theta f_{\theta_0}(x), \nabla_\theta f_{\theta_0}(y) \rangle \]
Kernels for deep models: neural tangent kernels

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \quad m \to \infty$$

**Neural tangent kernels** (NTK, Jacot et al., 2018)

- $\theta = (v_i, w_i)_i$, initialization $\theta_0 \sim N(0, I)$
- **Lazy training** (Chizat et al., 2019): $\theta$ stays close to $\theta_0$ when training with large $m$

$$f_\theta(x) \approx f_{\theta_0}(x) + \langle \theta - \theta_0, \nabla_\theta f_{\theta}(x) |_{\theta=\theta_0} \rangle.$$  

- Gradient descent for $m \to \infty \approx$ kernel ridge regression with **neural tangent kernel**

$$K_{NTK}(x, y) = \mathbb{E}_w[\sigma(w^\top x)\sigma(w^\top y) + (x^\top y)\sigma'(w^\top x)\sigma'(w^\top y)]$$

- with RELU networks, we obtain a dot-product kernel.
Kernels for deep models: dot-product kernels + Nyström

The Nyström method consists of replacing any point \( \varphi(x) \) in \( \mathcal{H} \), for \( x \) in \( \mathcal{X} \) by its orthogonal projection onto a finite-dimensional subspace

\[
\mathcal{F} = \text{span}(\varphi(z_1), \ldots, \varphi(z_p)),
\]

for some anchor points \( Z = [z_1, \ldots, z_p] \) in \( \mathbb{R}^{d \times p} \)
The projection is equivalent to

\[ \Pi_{\mathcal{F}}[x] := \sum_{j=1}^{p} \beta_j^* \varphi(z_j) \quad \text{with} \quad \beta^* \in \arg\min_{\beta \in \mathbb{R}^p} \left\| \varphi(x) - \sum_{j=1}^{p} \beta_j \varphi(z_j) \right\|_H^2, \]

Then, it is possible to show that with \( K(x, y) = \kappa(\langle x, y \rangle), \)

\[ K(x, y) \approx \langle \Pi_{\mathcal{F}}[x], \Pi_{\mathcal{F}}[y] \rangle_H = \langle \psi(x), \psi(y) \rangle_{\mathbb{R}^p}, \]

with

\[ \psi(x) = \kappa(Z^\top Z)^{-1/2} \kappa(Z^\top x), \]

where the function \( \kappa \) is applied pointwise to its arguments. The resulting \( \psi \) can be interpreted as a neural network performing (i) linear operation, (ii) pointwise non-linearity, (iii) linear operation.

(Williams and Seeger, 2001; Smola and Schölkopf, 2000; Fine and Scheinberg, 2001).
Nyström’s encoding with a dot-product kernel provides the encoding

\[ \psi_Z(x) = \kappa(Z^T Z)^{-1/2} \kappa(Z^T x). \]

The anchor points \( Z \) can be learned in various manners

- **unsupervised learning**: use K-means!
- **supervised learning**: use back-propagation

\[
\min_{w,Z} \frac{1}{n} \sum_{i=1}^{n} L(y_i, w^T \psi_Z(x_i)) + \lambda \|w\|^2.
\]
Kernels for deep models: end-to-end learning

Nyström’s encoding with a dot-product kernel provides the encoding

$$\psi_Z(x) = \kappa(Z^\top Z)^{-1/2}\kappa(Z^\top x).$$

The anchor points $Z$ can be learned in various manners

- **unsupervised learning**: use K-means!
- **supervised learning**: use back-propagation

$$\min_{w,Z} \frac{1}{n} \sum_{i=1}^{n} L(y_i, w^\top \psi_Z(x_i)) + \lambda\|w\|^2.$$  

end-to-end learning with kernels may mean learning a parametrized linear subspace of the RKHS, where we project the data.
Kernels for deep models: Convolutional Kernel Networks

What is the relation?

- it is possible to design functional spaces $\mathcal{H}$ where deep neural networks live (Mairal, 2016).

$$f(x) = \sigma_k(A_k\sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2\sigma_1(A_1x))\ldots)) = \langle f, \Phi(x) \rangle_{\mathcal{H}}.$$  

- we call the construction “convolutional kernel networks”.
Kernels for deep models: Convolutional Kernel Networks

What is the relation?

- it is possible to design functional spaces $\mathcal{H}$ where deep neural networks live (Mairal, 2016).

$$f(x) = \sigma_k(A_k \sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2 \sigma_1(A_1 x)) \ldots)) = \langle f, \Phi(x) \rangle_{\mathcal{H}}.$$

- we call the construction “convolutional kernel networks”.

Simple story about CKNs (Mairal, 2016)

- for the theory part, replace $x \mapsto \sigma(Ax)$ at each CNN layer by a kernel mapping $x \mapsto \varphi(x)$ associated to a dot-product kernel.

- for the practical part, replace $x \mapsto \sigma(Ax)$ by Nyström’s embedding $x \mapsto \kappa(Z^\top Z)^{-1/2} \kappa(Z^\top x)$. Then, you can either use K-means to learn the anchor points (unsupervised learning), or use back-propagation (supervised learning).
Kernels for deep models: Convolutional Kernel Networks

What is the relation?

- it is possible to design functional spaces $\mathcal{H}$ where deep neural networks live (Mairal, 2016).

$$f(x) = \sigma_k(A_k\sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2\sigma_1(A_1x)) \ldots)) = \langle f, \Phi(x) \rangle_\mathcal{H}.$$ 

- we call the construction “convolutional kernel networks”.

Why do we care?

- $\Phi(x)$ is related to the network architecture and is independent of training data. Is it stable? Does it lose signal information?
- $f$ is a predictive model. Can we control its stability?

$$|f(x) - f(x')| \leq \|f\|_\mathcal{H} \|\Phi(x) - \Phi(x')\|_\mathcal{H}.$$
Construction of the RKHS for continuous signals

Initial map $x_0$ in $L^2(\Omega, \mathcal{H}_0)$

$x_0 : \Omega \rightarrow \mathcal{H}_0$: **continuous** signal, with $\Omega = \mathbb{R}^d$ ($d = 2$ for images).

- $x_0(u) \in \mathcal{H}_0$: input value at location $u$ ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images).
Construction of the RKHS for continuous signals

Initial map $x_0$ in $L^2(\Omega, \mathcal{H}_0)$

$x_0 : \Omega \to \mathcal{H}_0$: \textbf{continuous} signal, with $\Omega = \mathbb{R}^d$ ($d = 2$ for images).

- $x_0(u) \in \mathcal{H}_0$: input value at location $u$ ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images).

Building map $x_k$ in $L^2(\Omega, \mathcal{H}_k)$ from $x_{k-1}$ in $L^2(\Omega, \mathcal{H}_{k-1})$

$x_k : \Omega \to \mathcal{H}_k$: \textbf{feature map} at layer $k$

\[ P_k x_{k-1}. \]

- $P_k$: \textbf{patch extraction} operator, extract small patch of feature map $x_{k-1}$ around each point $u$ ($P_k x_{k-1}(u)$ is a patch centered at $u$).
Construction of the RKHS for continuous signals

Initial map $x_0$ in $L^2(\Omega, \mathcal{H}_0)$

$x_0 : \Omega \rightarrow \mathcal{H}_0$: **continuous** signal, with $\Omega = \mathbb{R}^d$ ($d = 2$ for images).

- $x_0(u) \in \mathcal{H}_0$: input value at location $u$ ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images).

Building map $x_k$ in $L^2(\Omega, \mathcal{H}_k)$ from $x_{k-1}$ in $L^2(\Omega, \mathcal{H}_{k-1})$

$x_k : \Omega \rightarrow \mathcal{H}_k$: **feature map** at layer $k$

\[ M_k P_k x_{k-1}. \]

- $P_k$: **patch extraction** operator, extract small patch of feature map $x_{k-1}$ around each point $u$ ($P_k x_{k-1}(u)$ is a patch centered at $u$).

- $M_k$: **non-linear mapping** operator, maps each patch to a new Hilbert space $\mathcal{H}_k$ with a **pointwise** non-linear function $\varphi_k(\cdot)$.
Construction of the RKHS for continuous signals

Initial map $x_0$ in $L^2(\Omega, \mathcal{H}_0)$

$x_0 : \Omega \rightarrow \mathcal{H}_0$: continuous signal, with $\Omega = \mathbb{R}^d$ ($d = 2$ for images).

- $x_0(u) \in \mathcal{H}_0$: input value at location $u$ ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images).

Building map $x_k$ in $L^2(\Omega, \mathcal{H}_k)$ from $x_{k-1}$ in $L^2(\Omega, \mathcal{H}_{k-1})$

$x_k : \Omega \rightarrow \mathcal{H}_k$: feature map at layer $k$

$$x_k = A_k M_k P_k x_{k-1}.$$ 

- $P_k$: patch extraction operator, extract small patch of feature map $x_{k-1}$ around each point $u$ ($P_k x_{k-1}(u)$ is a patch centered at $u$).
- $M_k$: non-linear mapping operator, maps each patch to a new Hilbert space $\mathcal{H}_k$ with a pointwise non-linear function $\varphi_k(\cdot)$.
- $A_k$: (linear) pooling operator at scale $\sigma_k$. 
Construction of the RKHS for continuous signals

\[ x_k : \Omega \rightarrow \mathcal{H}_k \]

\[ x_{k-0.5} : \Omega \rightarrow \mathcal{H}_k \]

\[ x_{k-0.5}(v) = \varphi_k(P_k x_{k-1}(v)) \in \mathcal{H}_k \]

\[ P_k x_{k-1}(v) \in \mathcal{P}_k \text{ (patch extraction)} \]

\[ x_k(w) \in \mathcal{H}_k \]

\[ x_{k-1}(u) \in \mathcal{H}_{k-1} \]

\[ x_{k-1} : \Omega \rightarrow \mathcal{H}_{k-1} \]
Construction of the RKHS for continuous signals

Kernel mapping for patches

- We use a homogeneous dot-product kernel for image patches

\[ K(z, z') = \|z\| \|z'\| \kappa \left( \frac{\langle z, z' \rangle}{\|z\| \|z'\|} \right). \]

Multilayer representation

\[ \Phi_n(x) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in L^2(\Omega, \mathcal{H}_n). \]

- \( \sigma_k \) grows exponentially in practice (i.e., fixed with subsampling).
Construction of the RKHS for continuous signals

Kernel mapping for patches

- We use a homogeneous dot-product kernel for image patches

\[ K(z, z') = \|z\|\|z'\| \kappa \left( \frac{\langle z, z' \rangle}{\|z\|\|z'\|} \right). \]

Multilayer representation

\[ \Phi_n(x) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in \mathcal{L}^2(\Omega, \mathcal{H}_n). \]

- \( \sigma_k \) grows exponentially in practice (i.e., fixed with subsampling).

Prediction layer

- e.g., linear \( f(x) = \langle w, \Phi_n(x) \rangle. \)
- “linear kernel” \( \mathcal{K}(x, x') = \langle \Phi_n(x), \Phi_n(x') \rangle = \int_{\Omega} \langle x_n(u), x'_n(u) \rangle du. \)
Learning mechanism of CKNs between layers 0 and 1.
Convolutional Kernel Networks in Practice

What is the difference with a CNN?

- Given a patch $x$, a CNN computes $\psi_{CNN}(x) = \sigma(Z^T x)$.
- whereas a CKN computes $\psi_{CKN}(x) = \|x\| \kappa(Z^T Z)^{-1/2} \kappa(Z^T x/\|x\|)$.

Consequences:
- We have a geometric interpretation in terms of subspace learning.
- It provides unsupervised learning mechanisms (Nyström).
- Supervised learning is feasible.
- The kernel interpretation provides regularization mechanisms.
- Kernel representations can possibly be used in other contexts (statistical testing? kernel PCA? CCA? K-means?).
What is the difference with a CNN?

- Given a patch $x$, a CNN computes $\psi_{CNN}(x) = \sigma(Z^T x)$.
- whereas a CKN computes $\psi_{CKN}(x) = \|x\| \kappa (Z^T Z)^{-1/2} \kappa (Z^T x / \|x\|)$.

Consequences

- we have a geometric interpretation in terms of **subspace learning**.
- it provides unsupervised learning mechanisms (Nyström).
- supervised learning is feasible.
- the kernel interpretation provides regularization mechanisms.
- kernel representations can possibly be used in other contexts (statistical testing? kernel PCA? CCA? K-means?).
Experiments

- Briefly state-of-the-art for image retrieval (Paulin et al., 2015);
- Briefly state-of-the-art for image super-resolution (Mairal, 2016);

Interesting findings from CIFAR-10

- about 92% with supervision, mild data augmentation, 14 layers, 256 anchor points per layers (no need for batch norm, vanilla SGD+momentum).
- about 86% **with no supervision** for a two-layer model with a huge number of anchor points (1024-16384) and no data augmentation.
- with no supervision, the performance monotonically increases with the dimension (better kernel approximation).
- computing the exact kernel does not make sense in practice for computational reasons, but it is feasible with lots of CPUs; it yields about 90% with three layers (unpublished, by A. Bietti), which is consistent with (Shankar et al., 2020).
Other relations between kernels and deep learning

- hierarchical kernel descriptors (Bo et al., 2011);
- other multilayer models (Bouvrie et al., 2009; Montavon et al., 2011; Anselmi et al., 2015);
- deep Gaussian processes (Damianou and Lawrence, 2013).
- multilayer PCA (Schölkopf et al., 1998).
- old kernels for images (Scholkopf, 1997), related to one-layer CKN.
- RBF networks (Broomhead and Lowe, 1988).
- …
Outline

Open Problems and Research Topics
- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
- Foundations of deep learning from a kernel point of view
  - Motivation
  - Deep kernel machines
  - Deep learning and stability
  - Application to graphs
  - Application to biological sequences
Focus on convolutional kernel networks (CKNs)

What is the relation?

- it is possible to design functional spaces $\mathcal{H}$ for deep neural networks (Mairal, 2016).

\[
f(x) = \sigma_k(A_k\sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2\sigma_1(A_1x)) \ldots)) = \langle f, \Phi(x) \rangle_{\mathcal{H}}.
\]

- we call the construction “convolutional kernel networks” (in short, replace $u \mapsto \sigma(\langle a, u \rangle)$ by a kernel mapping $u \mapsto \varphi_k(u)$.

Why do we care?

- $\Phi(x)$ is related to the network architecture and is independent of training data. Is it stable? Does it lose signal information?
Focus on convolutional kernel networks (CKNs)

What is the relation?

- it is possible to design functional spaces $\mathcal{H}$ for deep neural networks (Mairal, 2016).

$$f(x) = \sigma_k(A_k \sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2 \sigma_1(A_1 x)) \ldots)) = \langle f, \Phi(x) \rangle_{\mathcal{H}}.$$  

- we call the construction “convolutional kernel networks” (in short, replace $u \mapsto \sigma(\langle a, u \rangle)$ by a kernel mapping $u \mapsto \varphi_k(u)$.

Why do we care?

- $\Phi(x)$ is related to the network architecture and is independent of training data. Is it stable? Does it lose signal information?
- $f$ is a predictive model. Can we control its stability?

$$|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \|\Phi(x) - \Phi(x')\|_{\mathcal{H}}.$$
Summary of the results from Bietti and Mairal (2019a)

Multi-layer construction of the RKHS $\mathcal{H}$

- Contains CNNs with smooth homogeneous activations functions.
Summary of the results from Bietti and Mairal (2019a)

Multi-layer construction of the RKHS $\mathcal{H}$
- Contains CNNs with smooth homogeneous activations functions.

Signal representation: Conditions for
- **Signal preservation** of the multi-layer kernel mapping $\Phi$.
- **Stability to deformations and non-expansiveness** for $\Phi$.
- Constructions to achieve **group invariance**.
Summary of the results from Bietti and Mairal (2019a)

Multi-layer construction of the RKHS $\mathcal{H}$
- Contains CNNs with smooth homogeneous activations functions.

Signal representation: Conditions for
- Signal preservation of the multi-layer kernel mapping $\Phi$.  
- Stability to deformations and non-expansiveness for $\Phi$.  
- Constructions to achieve group invariance.

On learning
- Bounds on the RKHS norm $\|\cdot\|_{\mathcal{H}}$ to control stability and generalization of a predictive model $f$.

\[ |f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \|\Phi(x) - \Phi(x')\|_{\mathcal{H}}. \]
Smooth homogeneous activations functions

\[
z \mapsto \text{ReLU}(w^\top z) \quad \implies \quad z \mapsto \|z\| \sigma\left(\frac{w^\top z}{\|z\|}\right).
\]
Stability to deformations

Deformations

- \( \tau : \Omega \rightarrow \Omega: \mathcal{C}^1 \)-diffeomorphism
- \( L_\tau x(u) = x(u - \tau(u)) \): action operator
- Much richer group of transformations than translations

- Studied for wavelet-based scattering transform (Mallat, 2012; Bruna and Mallat, 2013)
Stability to deformations

Deformations

- $\tau : \Omega \to \Omega$: $C^1$-diffeomorphism
- $L_\tau x(u) = x(u - \tau(u))$: action operator
- Much richer group of transformations than translations

Definition of stability

- Representation $\Phi(\cdot)$ is **stable** (Mallat, 2012) if:

  $$\| \Phi(L_\tau x) - \Phi(x) \| \leq (C_1 \| \nabla \tau \|_\infty + C_2 \| \tau \|_\infty) \| x \|$$

- $\| \nabla \tau \|_\infty = \sup_u || \nabla \tau(u) ||$ controls deformation
- $\| \tau \|_\infty = \sup_u | \tau(u) |$ controls translation
- $C_2 \to 0$: translation invariance
Smoothness and stability with kernels

**Geometry of the kernel mapping:** $f(x) = \langle f, \Phi(x) \rangle$

$$|f(x) - f(x')| \leq \|f\|_H \cdot \|\Phi(x) - \Phi(x')\|_H$$

- $\|f\|_H$ controls **complexity** of the model
- $\Phi(x)$ encodes CNN **architecture** independently of the model (smoothness, invariance, stability to deformations)
Smoothness and stability with kernels

**Geometry of the kernel mapping**: \( f(x) = \langle f, \Phi(x) \rangle \)

\[
|f(x) - f(x')| \leq \|f\|_H \cdot \|\Phi(x) - \Phi(x')\|_H
\]

- \( \|f\|_H \) controls **complexity** of the model
- \( \Phi(x) \) encodes CNN **architecture** independently of the model (smoothness, invariance, stability to deformations)

**Useful kernels in practice**:
- Convolutional kernel networks (**CKNs**, Mairal, 2016) with efficient approximations
- Extends to neural tangent kernels (**NTKs**, Jacot et al., 2018) of infinitely wide CNNs (Bietti and Mairal, 2019b)
Recap: Construction of the RKHS for continuous signals

\[ \Phi_n(x) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in L^2(\Omega, \mathcal{H}_n). \]
Patch extraction operator $P_k$

$$P_k x_{k-1}(u) := (x_{k-1}(u + v))_{v \in S_k} \in \mathcal{P}_k = \mathcal{H}_{k-1}^{S_k}$$

$x_{k-1}(u) \in \mathcal{H}_{k-1}$

$x_{k-1} : \Omega \to \mathcal{H}_{k-1}$

$P_k x_{k-1}(v) \in \mathcal{P}_k$ (patch extraction)
Patch extraction operator $P_k$

$$P_k x_{k-1}(u) := (x_{k-1}(u + v))_{v \in S_k} \in P_k = \mathcal{H}_{k-1}^{S_k}$$

- $S_k$: patch shape, e.g. box
Non-linear mapping operator $M_k$

$$M_k P_k x_{k-1}(u) := \varphi_k(P_k x_{k-1}(u)) \in \mathcal{H}_k$$
Non-linear mapping operator $M_k$

$$M_k P_{k-1}(u) := \varphi_k(P_{k-1}(u)) \in \mathcal{H}_k$$

Kernel mapping of **homogeneous dot-product kernels**:

$$K_k(z, z') = \|z\| \|z'\| \kappa_k\left(\frac{\langle z, z' \rangle}{\|z\| \|z'\|}\right) = \langle \varphi_k(z), \varphi_k(z') \rangle.$$  

$$\kappa_k(u) = \sum_{j=0}^{\infty} b_j u^j \text{ with } b_j \geq 0, \kappa_k(1) = 1$$

**Examples**

- $\kappa_{\text{exp}}(\langle z, z' \rangle) = e^{\langle z, z' \rangle} - 1$ (Gaussian kernel on the sphere)
- $\kappa_{\text{inv-poly}}(\langle z, z' \rangle) = \frac{1}{2 - \langle z, z' \rangle}$
Pooling operator $A_k$

$$x_k(u) = A_kM_kP_kx_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u - v)M_kP_kx_{k-1}(v)dv \in \mathcal{H}_k$$
Pooling operator $A_k$

\[ x_k(u) = A_k M_k P_k x_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u - v) M_k P_k x_{k-1}(v) dv \in \mathcal{H}_k \]

- $h_{\sigma_k}$: pooling filter at scale $\sigma_k$
- $h_{\sigma_k}(u) := \sigma_k^{-d} h(u/\sigma_k)$ with $h(u)$ Gaussian
- **linear, non-expansive operator**: $\|A_k\| \leq 1$
Pooling operator $A_k$

$$x_k(u) = A_k M_k P_k x_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u - v) M_k P_k x_{k-1}(v) dv \in \mathcal{H}_k$$

- $h_{\sigma_k}$: pooling filter at scale $\sigma_k$
- $h_{\sigma_k}(u) := \sigma_k^{-d} h(u/\sigma_k)$ with $h(u)$ Gaussian
- **linear, non-expansive operator**: $\|A_k\| \leq 1$
- In practice: **discretization**, sampling at resolution $\sigma_k$ after pooling
- “Preserves information” when **subsampling $\leq$ patch size**
Recap: $P_k$, $M_k$, $A_k$
Recap: multilayer construction

Multilayer representation

$$\Phi(x_0) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in L^2(\Omega, \mathcal{H}_n).$$

- $S_k, \sigma_k$ grow exponentially in practice (i.e., fixed with subsampling).
Recap: multilayer construction

Multilayer representation

\[ \Phi(x_0) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in L^2(\Omega, \mathcal{H}_n). \]

- \( S_k, \sigma_k \) grow exponentially in practice (i.e., fixed with subsampling).

Assumption on \( x_0 \)

- \( x_0 \) is typically a discrete signal acquired with physical device.
- Natural assumption: \( x_0 = A_0 x \), with \( x \) the original continuous signal, \( A_0 \) local integrator with scale \( \sigma_0 \) (anti-aliasing).
Recap: multilayer construction

Multilayer representation

\[ \Phi(x_0) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in L^2(\Omega, \mathcal{H}_n). \]

- \( S_k, \sigma_k \) grow exponentially in practice (i.e., fixed with subsampling).

Assumption on \( x_0 \)

- \( x_0 \) is typically a discrete signal acquired with physical device.
- Natural assumption: \( x_0 = A_0 x \), with \( x \) the original continuous signal, \( A_0 \) local integrator with scale \( \sigma_0 \) (anti-aliasing).

Final kernel

\[ K_{CKN}(x, x') = \langle \Phi(x), \Phi(x') \rangle_{L^2(\Omega)} = \int_{\Omega} \langle x_n(u), x'_n(u) \rangle du \]
Warmup: translation invariance

Representation

\[ \Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x. \]

How to achieve translation invariance?

- Translation: \( L_c x(u) = x(u - c) \).
Warmup: translation invariance

Representation

$$\Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x.$$ 

How to achieve translation invariance?

- Translation: $$L_c x(u) = x(u - c).$$
- *Equivariance* - all operators commute with $$L_c$$: $$\Box L_c = L_c \Box$$.

$$\parallel \Phi_n(L_c x) - \Phi_n(x) \parallel = \parallel L_c \Phi_n(x) - \Phi_n(x) \parallel$$

$$\leq \parallel L_c A_n - A_n \parallel \cdot \parallel M_n P_n \Phi_{n-1}(x) \parallel$$

$$\leq \parallel L_c A_n - A_n \parallel \parallel x \parallel.$$
Warmup: translation invariance

Representation

\[ \Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x. \]

How to achieve translation invariance?

- **Translation**: \( L_c x(u) = x(u - c) \).
- **Equivariance** - all operators commute with \( L_c \): \( \Box L_c = L_c \Box \).

\[
\| \Phi_n(L_c x) - \Phi_n(x) \| = \| L_c \Phi_n(x) - \Phi_n(x) \| \\
\leq \| L_c A_n - A_n \| \cdot \| M_n P_n \Phi_{n-1}(x) \| \\
\leq \| L_c A_n - A_n \| \| x \|. 
\]

- **Mallat (2012)**: \( \| L_\tau A_n - A_n \| \leq \frac{C_2}{\sigma_n} \| \tau \|_\infty \) (operator norm).
Warmup: translation invariance

Representation

$$\Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x.$$

How to achieve translation invariance?

- **Translation:** $$L_c x(u) = x(u - c).$$
- **Equivariance** - all operators commute with $$L_c$$: $$\square L_c = L_c \square$$.

$$\|\Phi_n(L_c x) - \Phi_n(x)\| = \|L_c \Phi_n(x) - \Phi_n(x)\|$$

$$\leq \|L_c A_n - A_n\| \cdot \|M_n P_n \Phi_{n-1}(x)\|$$

$$\leq \|L_c A_n - A_n\| \|x\|.$$

- Mallat (2012): $$\|L_c A_n - A_n\| \leq \frac{C_2}{\sigma_n} c$$ (operator norm).
- **Scale** $$\sigma_n$$ of the last layer controls translation invariance.
Stability to deformations

Representation

\[ \Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x. \]

How to achieve stability to deformations?

- Patch extraction \( P_k \) and pooling \( A_k \) do not commute with \( L_\tau \)!
Stability to deformations

Representation

\[ \Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x. \]

How to achieve stability to deformations?

- Patch extraction \( P_k \) and pooling \( A_k \) **do not commute** with \( L_\tau \! \)!
- \[ \| A_k L_\tau - L_\tau A_k \| \leq C_1 \| \nabla \tau \|_\infty \] (from Mallat, 2012).
Stability to deformations

Representation

\[ \Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x. \]

How to achieve stability to deformations?

- Patch extraction \( P_k \) and pooling \( A_k \) do not commute with \( L_\tau \)!
- \[ \|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty \] (from Mallat, 2012).
Stability to deformations

Representation

\[ \Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x. \]

How to achieve stability to deformations?

- Patch extraction \( P_k \) and pooling \( A_k \) do not commute with \( L_\tau \)!
- \( \| [A_k, L_\tau] \| \leq C_1 \| \nabla \tau \|_\infty \) (from Mallat, 2012).
- But: \( [P_k, L_\tau] \) is unstable at high frequencies!
Stability to deformations

Representation

\[ \Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x. \]

How to achieve stability to deformations?

- Patch extraction \( P_k \) and pooling \( A_k \) do not commute with \( L_\tau \)!
- \( \| [A_k, L_\tau] \| \leq C_1 \| \nabla \tau \|_\infty \) (from Mallat, 2012).
- But: \( [P_k, L_\tau] \) is unstable at high frequencies!
- Adapt to current layer resolution, patch size controlled by \( \sigma_{k-1} \):

\[ \| [P_k A_{k-1}, L_\tau] \| \leq C_{1,\kappa} \| \nabla \tau \|_\infty \sup_{u \in S_k} |u| \leq \kappa \sigma_{k-1} \]
Stability to deformations

Representation

\[ \Phi_n(x) := A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x. \]

How to achieve stability to deformations?

- Patch extraction \( P_k \) and pooling \( A_k \) do not commute with \( L_\tau \)!
- \( \| [A_k, L_\tau] \| \leq C_1 \| \nabla \tau \|_\infty \) (from Mallat, 2012).
- But: \( [P_k, L_\tau] \) is unstable at high frequencies!
- Adapt to current layer resolution, patch size controlled by \( \sigma_{k-1} \):

\[ \| [P_k A_{k-1}, L_\tau] \| \leq C_{1, \kappa} \| \nabla \tau \|_\infty \quad \sup_{u \in S_k} |u| \leq \kappa \sigma_{k-1} \]

- \( C_{1, \kappa} \) grows as \( \kappa^{d+1} \) \( \Rightarrow \) more stable with small patches (e.g., 3x3, VGG et al.).
Stability to deformations

**Theorem (Stability of CKN (Bietti and Mairal, 2019a))**

Let \( \Phi_n(x) = \Phi(A_0x) \) and assume \( \|\nabla \tau\|_\infty \leq 1/2 \),

\[
\|\Phi_n(L_\tau x) - \Phi_n(x)\| \leq \left( C_\beta (n + 1) \|\nabla \tau\|_\infty + \frac{C}{\sigma_n} \|\tau\|_\infty \right) \|x\|
\]

- Translation invariance: large \( \sigma_n \)
- Stability: small patch sizes (\( \beta \approx \) patch size, \( C_\beta = O(\beta^3) \) for images)
- Signal preservation: subsampling factor \( \approx \) patch size

\[\implies \text{need several layers with small patches} \quad n = O(\log(\sigma_n/\sigma_0)/\log \beta)\]
Stability to deformations

**Theorem (Stability of CKN (Bietti and Mairal, 2019a))**

Let \( \Phi_n(x) = \Phi(A_0x) \) and assume \( \| \nabla \tau \|_\infty \leq 1/2 \),

\[
\| \Phi_n(L_\tau x) - \Phi_n(x) \| \leq \left( C_\beta (n+1) \| \nabla \tau \|_\infty + \frac{C}{\sigma_n} \| \tau \|_\infty \right) \| x \|
\]

- Translation invariance: large \( \sigma_n \)
- Stability: small patch sizes (\( \beta \approx \) patch size, \( C_\beta = O(\beta^3) \) for images)
- Signal preservation: subsampling factor \( \approx \) patch size

\[ \implies \text{need several layers with small patches} \quad n = O(\log(\sigma_n/\sigma_0)/\log \beta) \]

- Achieved by controlling norm of **commutator** \([L_\tau, P_k A_{k-1}]\)
  - Extend result by Mallat (2012) for controlling \( \|[L_\tau, A]\| \)
  - Need patches \( S_k \) adapted to resolution \( \sigma_{k-1} \): \( \text{diam } S_k \leq \beta \sigma_{k-1} \)
Beyond the translation group

Can we achieve invariance to other groups?

- Group action: $L_g x(u) = x(g^{-1}u)$ (e.g., rotations, reflections).
- Feature maps $x(u)$ defined on $u \in G$ ($G$: locally compact group).
Beyond the translation group

Can we achieve invariance to other groups?

- Group action: \( L_g x(u) = x(g^{-1}u) \) (e.g., rotations, reflections).
- Feature maps \( x(u) \) defined on \( u \in G \) (\( G \): locally compact group).

Recipe: Equivariant inner layers + global pooling in last layer

- **Patch extraction:**
  \[ P_x(u) = (x(uv))_{v \in S}. \]

- **Non-linear mapping:** equivariant because pointwise!

- **Pooling** (\( \mu \): left-invariant Haar measure):
  \[ A_x(u) = \int_G x(uv) h(v) d\mu(v) = \int_G x(v) h(u^{-1}v) d\mu(v). \]

related work (Sifre and Mallat, 2013; Cohen and Welling, 2016; Raj et al., 2016)...
Stability to deformations for convolutional NTK

Theorem (Stability of NTK (Bietti and Mairal, 2019b))

Let $\Phi_n(x) = \Phi_{NTK}(A_0 x)$, and assume $\|\nabla \tau\|_\infty \leq 1/2$

$$\|\Phi_n(L_\tau x) - \Phi_n(x)\| \leq \left( C_\beta n^{7/4} \|\nabla \tau\|_\infty^{1/2} + C'_\beta n^2 \|\nabla \tau\|_\infty + \sqrt{n + \frac{1}{\sigma_n}} \|\tau\|_\infty \right) \|x\|,$$
Discretization and signal preservation: example in 1D

- Discrete signal $\tilde{x}_k$ in $\ell^2(\mathbb{Z}, \tilde{H}_k)$ vs continuous ones $x_k$ in $L^2(\mathbb{R}, \mathcal{H}_k)$.
- $\tilde{x}_k$: subsampling factor $s_k$ after pooling with scale $\sigma_k \approx s_k$:

$$\tilde{x}_k[n] = \bar{A}_k \bar{M}_k \bar{P}_k \bar{x}_{k-1}[ns_k].$$
Discretization and signal preservation: example in 1D

- Discrete signal $\bar{x}_k$ in $\ell^2(\mathbb{Z}, \mathcal{H}_k)$ vs continuous ones $x_k$ in $L^2(\mathbb{R}, \mathcal{H}_k)$.
- $\bar{x}_k$: subsampling factor $s_k$ after pooling with scale $\sigma_k \approx s_k$:
  $\bar{x}_k[n] = \bar{A}_k \bar{M}_k \bar{P}_k \bar{x}_{k-1}[ns_k]$.

- Claim: We can recover $\bar{x}_{k-1}$ from $\bar{x}_k$ if factor $s_k \leq$ patch size.
Discretization and signal preservation: example in 1D

- Discrete signal $\bar{x}_k$ in $\ell^2(\mathbb{Z}, \bar{H}_k)$ vs continuous ones $x_k$ in $L^2(\mathbb{R}, \mathcal{H}_k)$.
- $\bar{x}_k$: subsampling factor $s_k$ after pooling with scale $\sigma_k \approx s_k$:

  $$\bar{x}_k[n] = \bar{A}_k \bar{M}_k \bar{P}_k \bar{x}_{k-1}[ns_k].$$

- Claim: We can recover $\bar{x}_{k-1}$ from $\bar{x}_k$ if factor $s_k \leq$ patch size.
- How? Recover patches with linear functions (contained in $\bar{H}_k$)

  $$\langle f_w, \bar{M}_k \bar{P}_k \bar{x}_{k-1}(u) \rangle = f_w(\bar{P}_k \bar{x}_{k-1}(u)) = \langle w, \bar{P}_k \bar{x}_{k-1}(u) \rangle,$$

  and

  $$\bar{P}_k \bar{x}_{k-1}(u) = \sum_{w \in B} \langle f_w, \bar{M}_k \bar{P}_k \bar{x}_{k-1}(u) \rangle w.$$
Discretization and signal preservation: example in 1D

- Discrete signal $\bar{x}_k$ in $\ell^2(\mathbb{Z}, \bar{H}_k)$ vs continuous ones $x_k$ in $L^2(\mathbb{R}, H_k)$.
- $\bar{x}_k$: subsampling factor $s_k$ after pooling with scale $\sigma_k \approx s_k$:
  \[
  \bar{x}_k[n] = \bar{A}_k \bar{M}_k \bar{P}_k \bar{x}_{k-1}[ns_k].
  \]

- **Claim**: We can recover $\bar{x}_{k-1}$ from $\bar{x}_k$ if factor $s_k \leq$ patch size.
- **How?** Recover patches with **linear functions** (contained in $\bar{H}_k$)
  \[
  \langle f_w, \bar{M}_k \bar{P}_k \bar{x}_{k-1}(u) \rangle = f_w(\bar{P}_k \bar{x}_{k-1}(u)) = \langle w, \bar{P}_k \bar{x}_{k-1}(u) \rangle,
  \]
  and
  \[
  \bar{P}_k \bar{x}_{k-1}(u) = \sum_{w \in B} \langle f_w, \bar{M}_k \bar{P}_k \bar{x}_{k-1}(u) \rangle w.
  \]

**Warning**: no claim that recovery is practical and/or stable.
Discretization and signal preservation: example in 1D

\[ \bar{x}_{k-1} \]

deconvolution

\[ \bar{A}_k \bar{x}_{k-1} \]

recovery with linear measurements

\[ \bar{x}_k \]

downsampling

\[ \bar{A}_k \bar{M}_k \bar{P}_k \bar{x}_{k-1} \]

linear pooling

\[ \bar{M}_k \bar{P}_k \bar{x}_{k-1} \]

dot-product kernel

\[ \bar{x}_{k-1} \]

\[ \bar{P}_k \bar{x}_{k-1}(u) \in \mathcal{P}_k \]
RKHS of patch kernels $K_k$

$$K_k(z, z') = \|z\|\|z'\| \kappa\left(\frac{\langle z, z' \rangle}{\|z\|\|z'\|}\right), \quad \kappa(u) = \sum_{j=0}^{\infty} b_j u^j.$$

What does the RKHS contain?

Homogeneous version of (Zhang et al., 2016, 2017)
RKHS of patch kernels $K_k$

$$K_k(z, z') = \|z\|\|z'\|\kappa\left(\frac{\langle z, z' \rangle}{\|z\|\|z'\|}\right), \quad \kappa(u) = \sum_{j=0}^{\infty} b_j u^j.$$ 

What does the RKHS contain?

- RKHS contains **homogeneous functions**:

  $$f: z \mapsto \|z\|\sigma\left(\frac{\langle g, z \rangle}{\|z\|}\right).$$

Homogeneous version of (Zhang et al., 2016, 2017)
RKHS of patch kernels $K_k$

\[
K_k(z, z') = \|z\|\|z'\|\kappa \left( \frac{\langle z, z' \rangle}{\|z\|\|z'\|} \right), \quad \kappa(u) = \sum_{j=0}^{\infty} b_j u^j.
\]

What does the RKHS contain?

- **RKHS contains homogeneous functions:**
  \[
  f : z \mapsto \|z\|\sigma (\langle g, z \rangle / \|z\|).
  \]

- **Smooth activations:** $\sigma(u) = \sum_{j=0}^{\infty} a_j u^j$ with $a_j \geq 0$.

- **Norm:** $\|f\|_{H_k}^2 \leq C^2_\sigma(\|g\|^2) = \sum_{j=0}^{\infty} \frac{a_j^2}{b_j} \|g\|^2 < \infty$.

Homogeneous version of (Zhang et al., 2016, 2017)
RKHS of patch kernels $K_k$

Examples:

- $\sigma(u) = u$ (linear): $C_\sigma^2(\lambda^2) = O(\lambda^2)$.
- $\sigma(u) = u^p$ (polynomial): $C_\sigma^2(\lambda^2) = O(\lambda^{2p})$.
- $\sigma \approx \sin$, sigmoid, smooth ReLU: $C_\sigma^2(\lambda^2) = O(e^{c\lambda^2})$. 

![Graph showing RKHS examples](image)
Constructing a CNN in the RKHS $\mathcal{H}_K$

Some CNNs live in the RKHS: “linearization” principle

$$f(x) = \sigma_k(A_k\sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2\sigma_1(A_1x)) \ldots)) = \langle f, \Phi(x) \rangle_{\mathcal{H}}.$$
Constructing a CNN in the RKHS $H_K$

Some CNNs live in the RKHS: “linearization” principle

$$f(x) = \sigma_k(A_k\sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2\sigma_1(A_1x)) \ldots)) = \langle f, \Phi(x) \rangle_H.$$  

- Consider a CNN with filters $W_{ij}^k(u), u \in S_k$.
  - $k$: layer;
  - $i$: index of filter;
  - $j$: index of input channel.
- “Smooth homogeneous” activations $\sigma$.
- The CNN can be constructed hierarchically in $H_K$.
- Norm (linear layers):
  $$\|f_\sigma\|^2 \leq \|W_{n+1}\|_2^2 \cdot \|W_n\|_2^2 \cdot \|W_{n-1}\|_2^2 \ldots \|W_1\|_2^2.$$  
- Linear layers: product of spectral norms.
Link with generalization

Direct application of classical generalization bounds

- Simple bound on Rademacher complexity for linear/kernel methods:

\[ \mathcal{F}_B = \{ f \in \mathcal{H}_K, \| f \| \leq B \} \implies \text{Rad}_N(\mathcal{F}_B) \leq O \left( \frac{BR}{\sqrt{N}} \right). \]
Link with generalization

Direct application of classical generalization bounds

- Simple bound on Rademacher complexity for linear/kernel methods:

\[ \mathcal{F}_B = \{ f \in \mathcal{H}_K, \| f \| \leq B \} \implies \text{Rad}_N(\mathcal{F}_B) \leq O\left( \frac{BR}{\sqrt{N}} \right). \]

- Leads to margin bound \( O(\| \hat{f}_N \| R / \gamma \sqrt{N}) \) for a learned CNN \( \hat{f}_N \) with margin (confidence) \( \gamma > 0 \).

- Related to recent generalization bounds for neural networks based on product of spectral norms (e.g., Bartlett et al., 2017; Neyshabur et al., 2018).

(see, e.g., Boucheron et al., 2005; Shalev-Shwartz and Ben-David, 2014)
Deep convolutional representations: conclusions

Study of generic properties of signal representation

- **Deformation stability** with small patches, adapted to resolution.
- **Signal preservation** when subsampling ≤ patch size.
- **Group invariance** by changing patch extraction and pooling.
Deep convolutional representations: conclusions

Study of generic properties of signal representation

- **Deformation stability** with small patches, adapted to resolution.
- **Signal preservation** when subsampling $\leq$ patch size.
- **Group invariance** by changing patch extraction and pooling.

Applies to learned models

- Same quantity $\|f\|$ controls stability and generalization.
- “higher capacity” is needed to discriminate small deformations.
Deep convolutional representations: conclusions

Study of generic properties of signal representation
  - **Deformation stability** with small patches, adapted to resolution.
  - **Signal preservation** when subsampling $\leq$ patch size.
  - **Group invariance** by changing patch extraction and pooling.

Applies to learned models
  - Same quantity $\|f\|$ controls stability and generalization.
  - “higher capacity” is needed to discriminate small deformations.

Questions:
  - Better regularization?
  - How does SGD control capacity in CNNs?
  - What about networks with no pooling layers? ResNet?
Outline

6 Open Problems and Research Topics

- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
- Foundations of deep learning from a kernel point of view
  - Motivation
  - Deep kernel machines
  - Deep learning and stability
- Application to graphs
- Application to biological sequences
Graph-structured data is everywhere

(a) molecules

Propanal
(E)/(Z)-1-Propenol
Acetone
Acetaldehyde
Vinyl Alcohol
Ethylene Oxide
Aldehyde
Enol

(b) protein regulation

(c) social networks

(d) chemical pathways
Learning graph representations

**State-of-the-art models** for representing graphs:

- **Deep learning for graphs**: graph neural networks (GNNs);
- **Graph kernels**: Weisfeiler-Lehman (WL) graph kernels;
- **Hybrid models** attempt to bridge both worlds: graph neural tangent kernels (GNTK).
Learning graph representations

State-of-the-art models for representing graphs:

- **Deep learning for graphs**: graph neural networks (GNNs);
- **Graph kernels**: Weisfeiler-Lehman (WL) graph kernels;
- **Hybrid models** attempt to bridge both worlds: graph neural tangent kernels (GNTK).

Our model:

- A new type of **multilayer** graph kernel: more **expressive** than WL kernels;
- Learning easy-to-regularize and scalable **unsupervised** graph representations;
- Learning **supervised** graph representations like GNNs.
A graph is defined as a triplet $(\mathcal{V}, \mathcal{E}, a)$; 
- $\mathcal{V}$ and $\mathcal{E}$ correspond to the set of vertices and edges; 
- $a : \mathcal{V} \rightarrow \mathbb{R}^d$ is a function assigning attributes to each node.
Graph kernel mappings

- Map each graph $G$ in $\mathcal{X}$ to a vector $\Phi(G)$ in $\mathcal{H}$, which lends itself to learning tasks.
- A large class of graph kernel mappings can be written in the form

$$
\Phi(G) := \sum_{u \in \mathcal{V}} \varphi_{\text{base}}(\ell_G(u)) \quad \text{where } \varphi_{\text{base}} \text{ embeds some local patterns } \ell_G(u)
$$

(Shervashidze et al., 2011; Lei et al., 2017; Kriege et al., 2019)
Graph kernel mappings

- Map each graph $G$ in $\mathcal{X}$ to a vector $\Phi(G)$ in $\mathcal{H}$, which lends itself to learning tasks.
- A large class of graph kernel mappings can be written in the form

$$K(G, G') = \left\langle \sum_{u \in \mathcal{V}} \varphi_{\text{base}}(\ell_G(u)), \sum_{u' \in \mathcal{V}'} \varphi_{\text{base}}(\ell_{G'}(u')) \right\rangle.$$

where $\Phi(G)$ and $\Phi(G')$ are the vector representations of $G$ and $G'$, respectively.
Map each graph $G$ in $\mathcal{X}$ to a vector $\Phi(G)$ in $\mathcal{H}$, which lends itself to learning tasks.

A large class of graph kernel mappings can be written in the form

$$K(G, G') = \sum_{u \in \mathcal{V}} \sum_{u' \in \mathcal{V}'} \langle \varphi_{\text{base}}(\ell_G(u)), \varphi_{\text{base}}(\ell_{G'}(u')) \rangle.$$
Graph kernel mappings

- Map each graph $G$ in $\mathcal{X}$ to a vector $\Phi(G)$ in $\mathcal{H}$, which lends itself to learning tasks.
- A large class of graph kernel mappings can be written in the form

$$K(G, G') = \sum_{u \in \mathcal{V}} \sum_{u' \in \mathcal{V}'} \kappa_{\text{base}}(\ell_G(u), \ell_{G'}(u')).$$
Basic kernels: walk and path kernel mappings

Path kernels are more expressive than walk kernels, but less preferred for computational reasons.
Basic kernels: walk and path kernel mappings

- \( \mathcal{P}_k(G, u) \) := paths of length \( k \) from node \( u \) in \( G \). The \( k \)-path mapping is

\[
\varphi_{\text{path}}(u) := \sum_{p \in \mathcal{P}_k(G, u)} \delta_{a(p)} \implies \Phi(G) = \sum_{u \in \mathcal{V}} \sum_{p \in \mathcal{P}_k(G, u)} \delta_{a(p)}.
\]

- \( a(p) \): concatenated attributes in \( p \); \( \delta \): the Dirac function;
- \( \Phi(G) \) can be interpreted as a \textbf{histogram} of paths occurrences;
A relaxed path kernel

\[ \varphi_{\text{path}}(u) = \sum_{p \in \mathcal{P}_k(G, u)} \delta_a(p)(\cdot) \]

Issues of the path kernel mapping:
- \( \delta \) allows hard comparison between paths thus only works for discrete attributes;
- \( \delta \) is not differentiable, which cannot be “optimized” with back-propagation.
A relaxed path kernel

\[ \varphi_{\text{path}}(u) = \sum_{p \in \mathcal{P}_k(G,u)} \delta_{a(p)}(\cdot) \]

\[ \Rightarrow \sum_{p \in \mathcal{P}_k(G,u)} e^{-\frac{\alpha}{2} \|a(p) - \cdot\|^2} . \]

Issues of the path kernel mapping:
- \( \delta \) allows hard comparison between paths thus only works for discrete attributes;
- \( \delta \) is not differentiable, which cannot be “optimized” with back-propagation.

Relax it with a “soft” and differentiable mapping
- interpreted as the sum of Gaussians centered at each path from \( u \).
We define the one-layer GCKN as the relaxed path kernel mapping

\[ \varphi_1(u) := \sum_{p \in \mathcal{P}_k(G,u)} e^{-\frac{\alpha_1}{2} \|a(p) - \cdot\|^2} = \sum_{p \in \mathcal{P}_k(G,u)} \varphi_{RBF}(a(p)) \in \mathcal{H}_1. \]

This formula can be divided into 3 steps:

- path extraction: enumerating all \( \mathcal{P}_k(G,u) \);
- kernel mapping: evaluating Gaussian embedding \( \varphi_{RBF} \) of path features;
- path aggregation: aggregating the path embeddings.
One-layer GCKN: a closer look at the relaxed path kernel

- We define the one-layer GCKN as the relaxed path kernel mapping

$$\varphi_1(u) := \sum_{p \in \mathcal{P}_k(G, u)} e^{-\frac{\alpha_1}{2} \| a(p) - \cdot \|^2} = \sum_{p \in \mathcal{P}_k(G, u)} \varphi_{\text{RBF}}(a(p)) \in \mathcal{H}_1.$$  

- This formula can be divided into 3 steps:
  - path extraction: enumerating all $\mathcal{P}_k(G, u)$;
  - kernel mapping: evaluating Gaussian embedding $\varphi_{\text{RBF}}$ of path features;
  - path aggregation: aggregating the path embeddings.

- We obtain a new graph with the same topology but different features

$$(\mathcal{V}, \mathcal{E}, a) \xrightarrow{\varphi_{\text{path}}} (\mathcal{V}, \mathcal{E}, \varphi_1).$$
Construction of one-layer GCKN

\[ \varphi_1(u) \in \mathcal{H}_1 \]

\[ \varphi_1(u) := \varphi_{\text{RBF}}(a(p_1)) + \varphi_{\text{RBF}}(a(p_2)) + \varphi_{\text{RBF}}(a(p_3)) \]

\[ (\mathcal{V}, \mathcal{E}, \varphi_1 : \mathcal{V} \rightarrow \mathcal{H}_1) \]

\[ a(u) \in \mathbb{R}^d \]

\[ (\mathcal{V}, \mathcal{E}, a : \mathcal{V} \rightarrow \mathbb{R}^d) \]
From one-layer to multilayer GCKN

- We can repeat applying $\phi_{\text{path}}$ to the new graph

\[
(V, E, a) \xrightarrow{\phi_{\text{path}}} (V, E, \varphi_1) \xrightarrow{\phi_{\text{path}}} (V, E, \varphi_2) \xrightarrow{\phi_{\text{path}}} \ldots \xrightarrow{\phi_{\text{path}}} (V, E, \varphi_j).
\]

- Final graph representation at layer $j$, $\Phi(G) = \sum_{u \in V} \varphi_j(u)$. 

Why is the multilayer model interesting?

- Applying $\phi_{\text{path}}$ once can capture $\text{GCKN-path}$;
- Applying twice can capture $\text{GCKN-subtree}$;
- Applying more times may capture higher-order structures?

Long paths cannot be enumerated due to computational complexity, yet multilayer model can capture long-range substructures.
We can repeat applying $\varphi_{\text{path}}$ to the new graph

\[
(V, \mathcal{E}, a) \xrightarrow{\varphi_{\text{path}}} (V, \mathcal{E}, \varphi_1) \xrightarrow{\varphi_{\text{path}}} (V, \mathcal{E}, \varphi_2) \xrightarrow{\varphi_{\text{path}}} \ldots \xrightarrow{\varphi_{\text{path}}} (V, \mathcal{E}, \varphi_j).
\]

Final graph representation at layer $j$, $\Phi(G) = \sum_{u \in V} \varphi_j(u)$.

Why is the multilayer model interesting?

- applying $\varphi_{\text{path}}$ once can capture paths: GCKN-path;
- applying twice can capture subtrees: GCKN-subtree;
- applying more times may capture higher-order structures?
- **Long paths** cannot be enumerated due to computational complexity, yet multilayer model can capture long-range substructures.
Scalable approximation of Gaussian kernel mapping

\[ \varphi_{\text{path}}(u) = \sum_{p \in \mathcal{P}_k(G,u)} \varphi_{\text{RBF}}(a(p)). \]

- \( \varphi_{\text{RBF}}(a(p)) = e^{-\frac{\alpha}{2} \|a(p) - \cdot\|^2} \in \mathcal{H} \) is infinite-dimensional;

(Chen et al., 2019a,b; Williams and Seeger, 2001)
Scalable approximation of Gaussian kernel mapping

$$\varphi_{\text{path}}(u) = \sum_{p \in \mathcal{P}_k(G,u)} \varphi_{\text{RBF}}(a(p)).$$

- $$\varphi_{\text{RBF}}(a(p)) = e^{-\frac{\alpha}{2} \|a(p) - \cdot\|^2} \in \mathcal{H}$$ is infinite-dimensional;

- **Nyström** provides a finite-dimensional approximation $$\Psi(a(p))$$ by orthogonally projecting $$\varphi_{\text{RBF}}(a(p))$$ onto some finite-dimensional subspace:

  $$\text{Span}(\varphi_{\text{RBF}}(z_1), \ldots, \varphi_{\text{RBF}}(z_q))$$ parametrized by $$Z = \{z_1, \ldots, z_q\},$$

where $$z_j \in \mathbb{R}^{dk}$$ can be interpreted as path features.

(Chen et al., 2019a,b; Williams and Seeger, 2001)
Scalable approximation of Gaussian kernel mapping

\[ \varphi_{\text{path}}(u) = \sum_{p \in \mathcal{P}_k(G,u)} \varphi_{\text{RBF}}(a(p)). \]

- \( \varphi_{\text{RBF}}(a(p)) = e^{-\frac{\alpha}{2} \|a(p) - \cdot\|^2} \in \mathcal{H} \) is infinite-dimensional;
- **Nyström** provides a **finite-dimensional** approximation \( \Psi(a(p)) \) by orthogonally projecting \( \varphi_{\text{RBF}}(a(p)) \) onto some finite-dimensional subspace:
  
  \[ \text{Span}(\varphi_{\text{RBF}}(z_1), \ldots, \varphi_{\text{RBF}}(z_q)) \]
  
  parametrized by \( Z = \{z_1, \ldots, z_q\} \),

where \( z_j \in \mathbb{R}^{dk} \) can be interpreted as path features.
- The parameters \( Z \) can be learned by
  - (unsupervised) K-means on the set of path features;
  - (supervised) end-to-end learning with back-propagation.

(Chen et al., 2019a,b; Williams and Seeger, 2001)
## Comparison of GCKN and GNN

<table>
<thead>
<tr>
<th>GCKN</th>
<th>vs.</th>
<th>GNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{\text{GCKN}}(G) = \sum_{u \in G} \psi_k(u) )</td>
<td></td>
<td>( f_{\text{GNN}}(G) = \sum_{u \in G} f_k(u) )</td>
</tr>
<tr>
<td>( \psi_k(u) = \sum_{p \in \mathcal{P}<em>k(G,u)} \kappa(Z^T Z)^{-\frac{1}{2}} \kappa(Z^T \psi</em>{k-1}(p)) )</td>
<td></td>
<td>( f_k(u) = \sum_{v \in \mathcal{N}(u)} \text{ReLU}(Z^T f_{k-1}(v)) )</td>
</tr>
<tr>
<td>local path aggregation</td>
<td>neighborhood aggregation</td>
<td>?</td>
</tr>
<tr>
<td>projection in a known RKHS</td>
<td>supervised and unsupervised</td>
<td>supervised</td>
</tr>
</tbody>
</table>
Experiments on graphs with discrete attributes

- Accuracy improvement with respect to the WL subtree kernel.
- GCKN-path already outperforms the baselines.
- Increasing number of layers brings larger improvement.
- Supervised learning does not improve performance, but leads to more compact representations.

(Shervashidze et al., 2011; Du et al., 2019; Xu et al., 2019; Kipf and Welling, 2017)
Experiments on graphs with continuous attributes

- Accuracy improvement with respect to the WWL kernel.
- Results similar to discrete case.
- Path features seem presumably predictive enough.

(Du et al., 2019; Togninalli et al., 2019)
Model interpretation for Mutagenicity prediction

- Idea: find the minimal connected component that preserves the prediction.

(Ying et al., 2019)
Open Problems and Research Topics

- Multiple Kernel Learning (MKL)
- Large-scale learning with kernels
- **Foundations of deep learning from a kernel point of view**
  - Motivation
  - Deep kernel machines
  - Deep learning and stability
  - Application to graphs
- Application to biological sequences
Sequence modeling as a supervised learning problem

\[
\min_{f \in F} \sum_{i=1}^{n} L(y_i, f(x_i)) + \mu \Omega(f)
\]

Empirical risk, data fit + regularization.

How do we define the functional space \( F \)?
Sequence modeling as a supervised learning problem

- Biological sequences $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and their associated labels $y_1, \ldots, y_n$.
- Goal: learning a **predictive** and **interpretable** function $f : \mathcal{X} \rightarrow \mathbb{R}$

$$
\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \mu \Omega(f) .
$$

- empirical risk, data fit
- regularization

- How do we define the functional space $\mathcal{F}$?
String kernels

A classical approach for modeling biological sequences over alphabet $\mathcal{A}$ relies on string kernels.

$$K(x, x') = \sum_{u \in \mathcal{A}^k} \delta_u(x) \delta_u(x'),$$

where $u$ is a $k$-mer over an alphabet $\mathcal{A}$ and $\delta_u(x)$ can be:

- the number of occurrences of $u$ in $x$: **spectrum kernel** (Leslie et al., 2002);
- the number of occurrences of $u$ in $x$ up to $m$ mismatches: **mismatch kernel** (Leslie and Kuang, 2004);
- the number of occurrences of $u$ in $x$ allowing gaps, with a weight decaying exponentially with the number of gaps: **substring kernel** (Lodhi et al., 2002).

What is $\Phi(x)$?

It can be interpreted as a histogram of pattern occurrences.
String kernels

A classical approach for modeling biological sequences over alphabet $\mathcal{A}$ relies on string kernels.

$$K(x, x') = \sum_{u \in \mathcal{A}^k} \delta_u(x)\delta_u(x') = \langle \Phi(x), \Phi(x') \rangle,$$

where $u$ is a $k$-mer over an alphabet $\mathcal{A}$ and $\delta_u(x)$ can be:

- the number of occurrences of $u$ in $x$: **spectrum kernel** (Leslie et al., 2002);
- the number of occurrences of $u$ in $x$ up to $m$ mismatches: **mismatch kernel** (Leslie and Kuang, 2004);
- the number of occurrences of $u$ in $x$ allowing gaps, with a weight decaying exponentially with the number of gaps: **substring kernel** (Lodhi et al., 2002).

**What is $\Phi(x)$?**

It can be interpreted as a histogram of pattern occurences.
Define a continuous relaxation of the mismatch kernel (Chen et al., 2019a; Morrow et al., 2017)

\[
K_{\text{CKN}}(x, x') = \sum_{i=1}^{\lfloor x\rfloor - k + 1} \sum_{j=1}^{\lfloor x'\rfloor - k + 1} K_0(x[i:i+k], x'[j:j+k]).
\]

- Use one-hot encoding

\[
x[i:i+5] := \text{TTGAG} \mapsto \begin{array}{c}
A \\
T \\
C \\
G
\end{array} \begin{bmatrix}
0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1
\end{bmatrix}.
\]

- \(K_0\) is a Gaussian kernel over one-hot representations of k-mers (in \(\mathbb{R}^{k \times d}\)).
Define a continuous relaxation of the mismatch kernel (Chen et al., 2019a; Morrow et al., 2017)

\[ K_{CKN}(x, x') = \sum_{i=1}^{\lvert x \rvert - k + 1} \sum_{j=1}^{\lvert x' \rvert - k + 1} \langle \varphi_0(x[i:i+k]), \varphi_0(x'[j:j+k]) \rangle. \]

- Use one-hot encoding

\[ x[i:i+5] := \text{TTGAG} \rightarrow \begin{bmatrix} A & T & C & G \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \ 1 & 1 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 1 & 0 & 1 \end{bmatrix}. \]

- \( K_0 \) is a Gaussian kernel over one-hot representations of k-mers (in \( \mathbb{R}^{k \times d} \)).
Define a continuous relaxation of the mismatch kernel (Chen et al., 2019a; Morrow et al., 2017)

\[
K_{\text{CKN}}(\mathbf{x}, \mathbf{x'}) = \left\langle \sum_{i=1}^{\|\mathbf{x}\|-k+1} \varphi_0(\mathbf{x}[i:i+k]), \sum_{j=1}^{\|\mathbf{x'}\|-k+1} \varphi_0(\mathbf{x'}[j:j+k]) \right\rangle .
\]

- Use one-hot encoding

\[
\begin{align*}
\mathbf{x}[i:i+5] := \text{TTGAG} & \mapsto \\
A & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
T & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \end{bmatrix} \\
C & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
G & \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \end{bmatrix}
\end{align*}
\]

- \(K_0\) is a Gaussian kernel over \textbf{one-hot} representations of k-mers (in \(\mathbb{R}^{k \times d}\)).
Scalable Approximation of Kernel Mapping (with more details this time)

\[ K_0(u, u') = \langle \varphi_0(u), \varphi_0(u') \rangle_{\mathcal{H}_0} \approx \langle \psi_0(u), \psi_0(u') \rangle_{\mathbb{R}^q}. \]

- **Nyström** provides a finite-dimensional approximation \( \psi_0(u) \) in \( \mathbb{R}^q \) by orthogonally projecting \( \varphi_0(u) \) onto some finite-dimensional subspace:

\[ \mathcal{E}_0 = \text{Span}(\varphi_0(z_1), \ldots, \varphi_0(z_q)) \] parametrized by \( Z = \{z_1, \ldots, z_q\} \).
Scalable Approximation of Kernel Mapping (with more details this time)

\[ K_0(u, u') = \langle \varphi_0(u), \varphi_0(u') \rangle_{\mathcal{H}_0} \approx \langle \psi_0(u), \psi_0(u') \rangle_{\mathbb{R}^q}. \]

- **Nyström** provides a **finite-dimensional** approximation \( \psi_0(u) \) in \( \mathbb{R}^q \) by orthogonally projecting \( \varphi_0(u) \) onto some finite-dimensional subspace:
  \[ \mathcal{E}_0 = \text{Span}(\varphi_0(z_1), \ldots, \varphi_0(z_q)) \text{ parametrized by } Z = \{z_1, \ldots, z_q\}. \]

- **General case:**
  \[ \psi_0(u) = K_0(Z, Z)^{-1/2} K_0(Z, u). \]
Scalable Approximation of Kernel Mapping (with more details this time)

\[ K_0(u, u') = \langle \varphi_0(u), \varphi_0(u') \rangle_{H_0} \approx \langle \psi_0(u), \psi_0(u') \rangle_{\mathbb{R}^q}. \]

- **Nyström** provides a **finite-dimensional** approximation \( \psi_0(u) \) in \( \mathbb{R}^q \) by orthogonally projecting \( \varphi_0(u) \) onto some finite-dimensional subspace:

\[ \mathcal{E}_0 = \text{Span}(\varphi_0(z_1), \ldots, \varphi_0(z_q)) \text{ parametrized by } Z = \{z_1, \ldots, z_q\}. \]

- **Case of dot-product kernels** \( K_0(u, u') = \kappa(\langle u, u' \rangle) \):

\[ \psi_0(u) = \kappa(Z^\top Z)^{-1/2} \kappa(Z^\top u). \]

linear operation - pointwise nonlinearity - linear operation (subject to interpretation)

Ex: \( \kappa(\beta) = e^{\beta-1} \), polynomial, inverse polynomial, arc-cosine kernels....
Single-Layer CKN for sequence modeling

\[ \psi_0(P_i(x)) = K^{-\frac{1}{2}}K_Z(P_i(x)) \]

\[ \langle w, \Psi(x) \rangle \]

Global pooling

Kernel mapping approximation

Prediction layer
Multilayer CKN for sequence modeling

\[ \mathbf{x}_k \in \mathcal{H}_k \]

\[ \vdots \]

\[ \mathbf{x}_1(v_3) \in \mathcal{H}_1 \]

\[ \mathbf{x}_1 : \Omega_1 \rightarrow \mathcal{H}_1 \]

\[ \mathbf{x}_{0.5} : \Omega_0 \rightarrow \mathcal{H}_1 \]

\[ \mathbf{x}_0.5(v_2) = \varphi_1(P_1\mathbf{x}_0(v_2)) \in \mathcal{H}_1 \]

\[ \mathbf{x}_0(v_1) = \varphi_0(P_0\mathbf{x}(v_1)) \in \mathcal{H}_0 \]

\[ \mathbf{x}_0 : \Omega_0 \rightarrow \mathcal{H}_0 \]

\[ \mathbf{x} : \Omega \rightarrow \mathcal{A} \]

\[ \mathbf{x}(u) \in \mathcal{A} \]

\[ P_0\mathbf{x}(v_1) \in \mathcal{P}_0 \text{ (patch)} \]

\[ P_1\mathbf{x}_0(v_2) \in \mathcal{P}_1 \]

\[ \text{domain-specific kernel} \]

\[ \text{dot-product kernel} \]

\[ \text{linear pooling} \]

\[ \text{embedding layer} \]

\[ \text{convolutional kernel layer} \]

\[ \text{prediction layer} \]
From k-mers to gapped k-mers

k-mers with gaps

- For a sequence $x = x_1 \ldots x_n \in \mathcal{X}$ of length $n$ and a sequence of ordered indices $i = (i_1, \ldots, i_k)$ in $\mathcal{I}(k, n)$, we define a k-substring as:

  $$x[i] = x_{i_1}x_{i_2}\ldots x_{i_k}.$$ 

- We introduce the quantity

  $$\text{gaps}(i) = \text{number of gaps in index sequence}.$$ 

- Example: $x = \text{ABRACADABRA}$

  $$i = (4, 5, 8, 9, 11) \quad x[i] = \text{RADAR} \quad \text{gaps}(i) = 3.$$
Recurrent kernel networks

Comparing all the k-mers between a pair of sequences (single layer models)

\[
K_{CKN}(x, x') = \sum_{i=1}^{\lfloor |x| - k + 1 \rfloor} \sum_{j=1}^{\lfloor |x'| - k + 1 \rfloor} K_0 \left( x[i:i+k], x'[j:j+k] \right).
\]

- The kernel mapping is \( \Phi(x) = \sum_{i=1}^{\lfloor |x| - k + 1 \rfloor} \varphi_0(x[i:i+k]) \).
Recurrent kernel networks

Comparing all the gapped k-mers between a pair of sequences (single layer models)

\[ K_{RKN}(x, x') = \sum_{i \in I(k, |x|)} \sum_{j \in I(k, |x'|)} \lambda \text{gaps}(i) \lambda \text{gaps}(j) K_0 \left( x_i, x'_j \right). \]

- The kernel mapping is \( \Phi(x) = \sum_{i \in I(k, |x|)} \lambda \text{gaps}(i) \varphi_0(x_i). \)
- This is a differentiable relaxation of the substring kernel.

But enumerating all possible substrings is costly...
Approximation and recursive computation of RKN

Approximate feature map of RKN kernel

The approximate feature map of $K_{RKN}$ via Nyström approximation is

$$\Psi(x) = \sum_{i \in I(k,t)} \lambda^{\text{gaps}(i)} \psi_0(x_{[i]}) \in \mathbb{R}^q,$$

where, as usual with a dot-product kernel,

$$\psi_0(x_{[i]}) = \kappa(Z^\top Z)^{-1/2}\kappa(Z^\top x_{[i]}).$$

- The sum can be computed by using dynamic programming (Lodhi et al., 2002),
- which leads to a particular recurrent neural network (see Lei et al., 2017).
A feature map for the single-layer RKN

When $K_0$ is a Gaussian kernel, the feature map of RKN is a mixture of Gaussians centered at $x[i]$, weighted by the corresponding penalization $\lambda_{\text{gaps}(i)}$.

$$\lambda^2 \varphi_0(x[i])$$

$$\sum_i \lambda_{\text{gap}(i)} \varphi_0(x[i])$$

Figure: Example of $K_{\text{RKN}}$ for $k = 4$
## Results

Protein fold classification on SCOP 2.06 (Hou et al., 2017) (using more informative sequence features including PSSM, secondary structure and solvent accessibility)

<table>
<thead>
<tr>
<th>Method</th>
<th>#Params</th>
<th>Accuracy</th>
<th>Level-stratified accuracy (top1/top5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>top 1</td>
<td>top 5 family</td>
</tr>
<tr>
<td>PSI-BLAST</td>
<td>-</td>
<td>84.53</td>
<td>86.48</td>
</tr>
<tr>
<td>DeepSF</td>
<td>920k</td>
<td>73.00</td>
<td>90.25</td>
</tr>
<tr>
<td>CKN (128 filters)</td>
<td>211k</td>
<td>76.30</td>
<td>92.17</td>
</tr>
<tr>
<td>CKN (512 filters)</td>
<td>843k</td>
<td>84.11</td>
<td>94.29</td>
</tr>
<tr>
<td>RKN (128 filters)</td>
<td>211k</td>
<td>77.82</td>
<td>92.89</td>
</tr>
<tr>
<td>RKN (512 filters)</td>
<td>843k</td>
<td><strong>85.29</strong></td>
<td><strong>94.95</strong></td>
</tr>
</tbody>
</table>

**Note:** More experiments with statistical tests have been conducted in our paper.

(Hou et al., 2017; Chen et al., 2019a)
Logos, by finding pre-image of each filter
## Results

**Protein fold recognition on SCOP 1.67** (widely used in the past)

<table>
<thead>
<tr>
<th>Method</th>
<th>pooling</th>
<th>one-hot auROC</th>
<th>one-hot auROC50</th>
<th>BLOSUM62 auROC</th>
<th>BLOSUM62 auROC50</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-pairwise</td>
<td></td>
<td>0.724</td>
<td>0.359</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mismatch</td>
<td></td>
<td>0.814</td>
<td>0.467</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LA-kernel</td>
<td>–</td>
<td>–</td>
<td></td>
<td>0.834</td>
<td>0.504</td>
</tr>
<tr>
<td>LSTM</td>
<td></td>
<td>0.830</td>
<td>0.566</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CKN</td>
<td></td>
<td>0.837</td>
<td>0.572</td>
<td>0.866</td>
<td>0.621</td>
</tr>
<tr>
<td>RKN</td>
<td>mean</td>
<td>0.829</td>
<td>0.541</td>
<td>0.840</td>
<td>0.571</td>
</tr>
<tr>
<td>RKN</td>
<td>max</td>
<td><strong>0.844</strong></td>
<td><strong>0.587</strong></td>
<td><strong>0.871</strong></td>
<td><strong>0.629</strong></td>
</tr>
<tr>
<td>RKN (unsup)</td>
<td>mean</td>
<td>0.805</td>
<td>0.504</td>
<td>0.833</td>
<td>0.570</td>
</tr>
</tbody>
</table>

(Liao and Noble, 2003; Leslie et al., 2003; Vert et al., 2004b; Hochreiter et al., 2007; Chen et al., 2019a)
Conclusion of the course
What we saw

- Basic definitions of p.d. kernels and RKHS
- How to use RKHS in machine learning
- The importance of the choice of kernels, and how to include “prior knowledge” there.
- Several approaches for kernel design (there are many!)
- Review of kernels for strings and on graphs
- Recent research topics about kernel methods
What we did not see

- How to automatize the process of kernel design (kernel selection? kernel optimization?)
- How to deal with non p.d. kernels
- Bayesian view of kernel methods, called Gaussian processes.
- How do statistical testing with kernels with the kernel mean embedding.


References VII


References


A. Nicholls. Oechem, version 1.3.4, openeye scientific software. website, 2005.


References XIV


B. Weisfeiler and A. A. Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia, Ser. 2*, 9, 1968.


References XV


