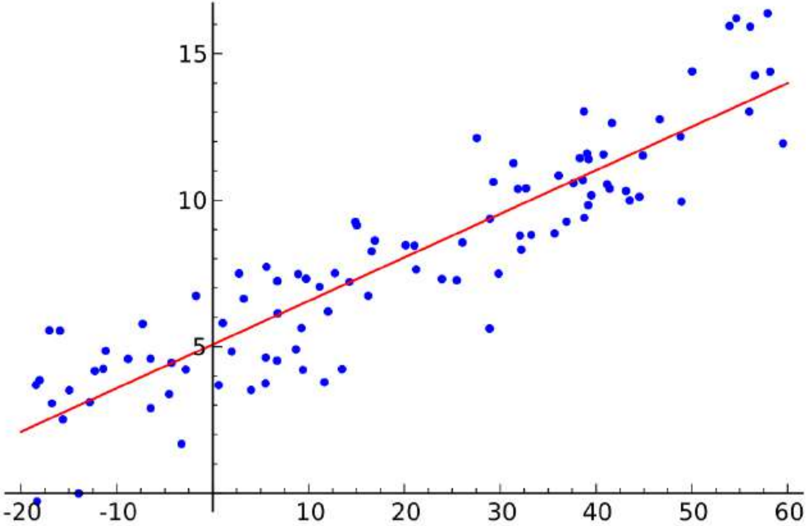
Machine Learning with Kernel Methods

Julien Mairal & Jean-Philippe Vert

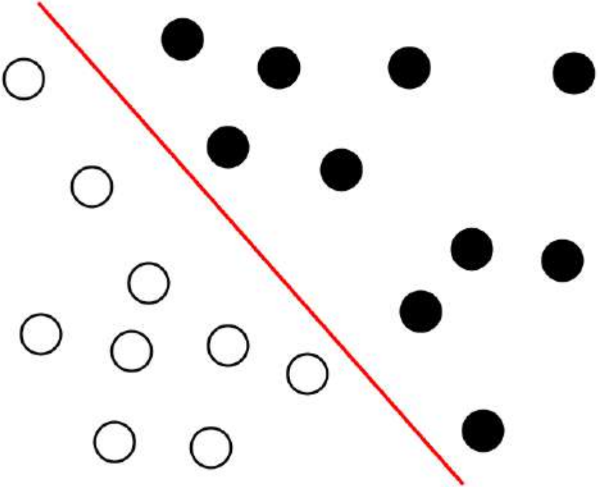
`firstname.lastname@m4x.org`

Last update: Dec 2020

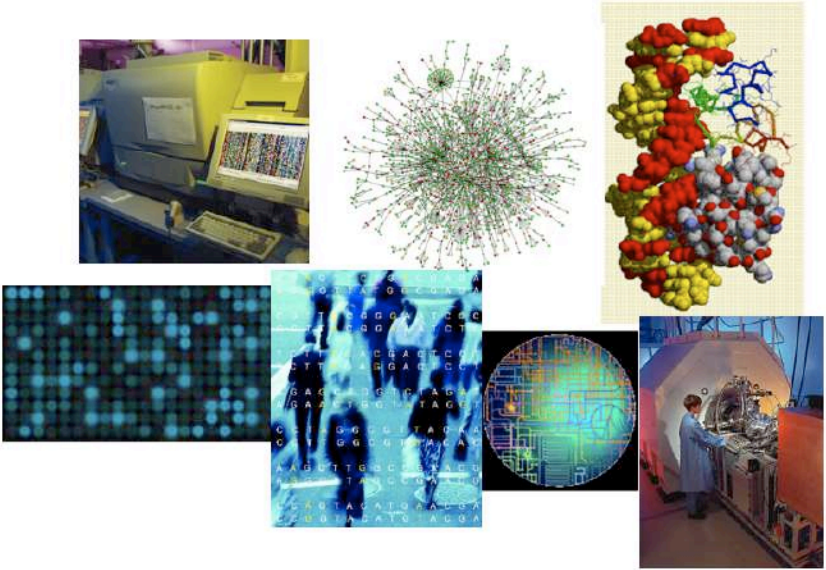
Starting point: what we know is how to solve



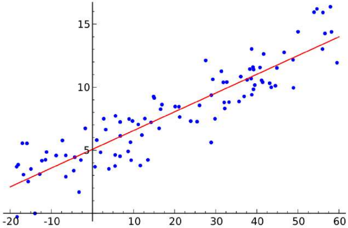
Or

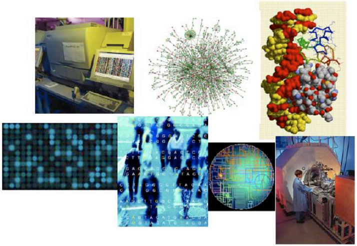


But real data are often more complicated...



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

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

Part 1

Kernels and RKHS

Overview

Motivations

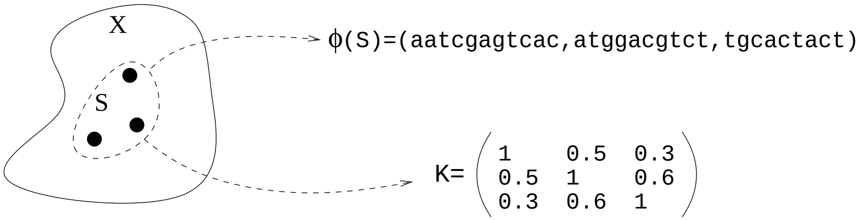
- Develop versatile algorithms to process and analyze data...
- ...without making any assumptions regarding the type of data (vectors, strings, graphs, images, ...)

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

Representation by pairwise comparisons



`ϕ(S)=(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 $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ by the $n \times n$ matrix:

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

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 $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is symmetric:

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

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

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

Similarity matrices of p.d. kernels

Remarks

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

The simplest p.d. kernel, for real numbers

Lemma

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

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

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:

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

is p.d.

Proof:

- $xx' = x'x$
- $\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j x_i x_j = \left(\sum_{i=1}^{N} a_i x_i\right)^2 \geq 0$ ☐

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:

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

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:

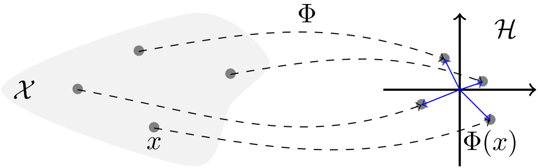
$$\forall \left(\mathbf{x}, \mathbf{x}'\right) \in \mathcal{X}^2, \quad K\left(\mathbf{x}, \mathbf{x}'\right) = \left\langle \mathbf{x}, \mathbf{x}' \right\rangle_{\mathbb{R}^d}$$

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

Proof:

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

A more ambitious p.d. kernel

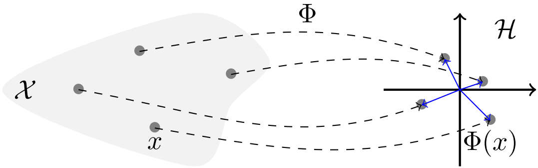


Lemma

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

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

A more ambitious p.d. kernel



Lemma

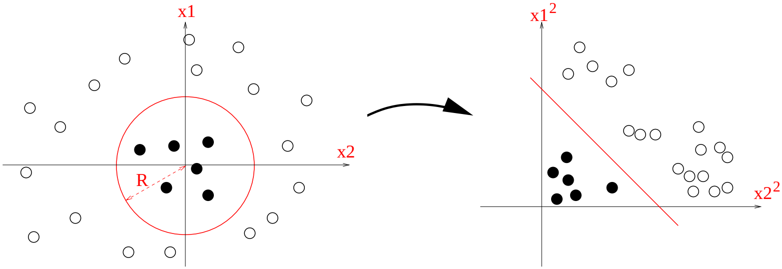
Let $\mathcal{X}$ be any set, and $\Phi : \mathcal{X} \mapsto \mathbb{R}^d$. Then, the function $K : \mathcal{X}^2 \mapsto \mathbb{R}$ defined as follows is p.d.:

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

Proof:

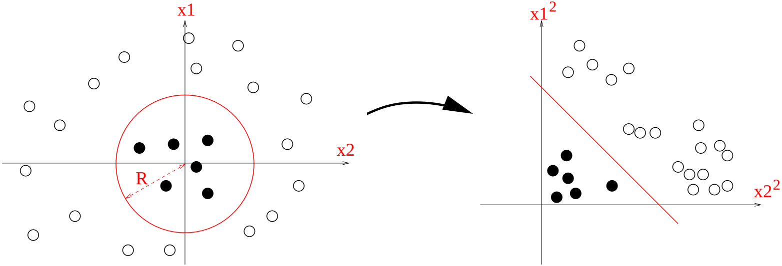
- $\langle \Phi\left(\mathbf{x}\right), \Phi\left(\mathbf{x}'\right) \rangle_{\mathbb{R}^d} = \langle \Phi\left(\mathbf{x}'\right), \Phi\left(\mathbf{x}\right) \rangle_{\mathbb{R}^d}$
- $\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j \langle \Phi\left(\mathbf{x}_i\right), \Phi\left(\mathbf{x}_j\right) \rangle_{\mathbb{R}^d}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

Example: polynomial kernel



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

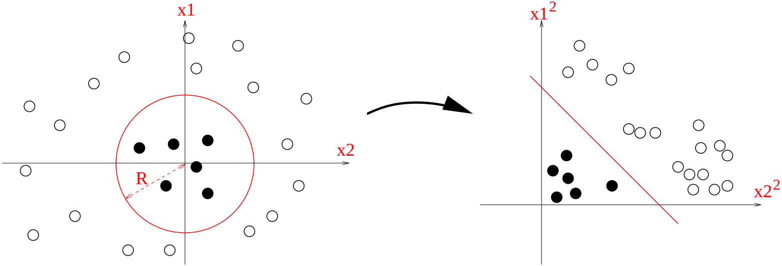
Example: polynomial kernel



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

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2 x_2'^2 \\ &= \left(x_1x_1' + x_2x_2'\right)^2 \\ &= \left\langle \mathbf{x}, \mathbf{x}' \right\rangle_{\mathbb{R}^2}^2 \quad . \end{aligned}$$

Example: polynomial kernel



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

$$
\begin{aligned}
K(\mathbf{x}, \mathbf{x}') &= x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 \\
&= \left(x_1x_1' + x_2x_2'\right)^2 \\
&= \left\langle \mathbf{x}, \mathbf{x}' \right\rangle_{\mathbb{R}^2}^2 \ .
\end{aligned}
$$

Exercise: show that $\langle \mathbf{x}.\mathbf{x}' \rangle_{\mathbb{R}^p}^d$ *is p.d. on* $\mathcal{X} = \mathbb{R}^p$ *for any* $d \in \mathbb{N}$.

Conversely: Kernels as inner products

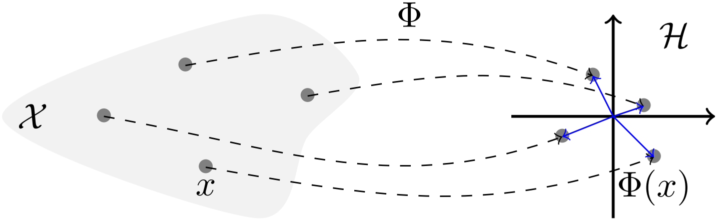
Theorem (Aronszajn, 1950)

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 $\mathbf{x}, \mathbf{x}'$ in $\mathcal{X}$:

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



In case of ...

Definitions

- An inner product on an $\mathbb{R}$-vector space $\mathcal{H}$ is a mapping $(f, g) \mapsto \langle f, g \rangle_{\mathcal{H}}$ from $\mathcal{H}^2$ to $\mathbb{R}$ that is bilinear, symmetric and such that $\langle f, f \rangle_{\mathcal{H}} > 0$ for all $f \in \mathcal{H} \backslash \{0\}$.
- 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 $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ 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 $[\mathbf{K}]_{ij} := K(\mathbf{x}_i, \mathbf{x}_j)$.
- It can therefore be diagonalized on an orthonormal basis of eigenvectors $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N)$, with non-negative eigenvalues $0 \leq \lambda_1 \leq \ldots \leq \lambda_N$, i.e.,

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

with

$$\Phi(\mathbf{x}_i) = \begin{pmatrix} \sqrt{\lambda_1}[\mathbf{u}_1]_i \\ \vdots \\ \sqrt{\lambda_N}[\mathbf{u}_N]_i \end{pmatrix} . \quad \square$$

Proof: general case

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

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

Outline

1. Kernels and RKHS
    - 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

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 $\mathbf{x}$ in $\mathcal{X}$ will be represented by a function $\Phi(\mathbf{x}) = K_{\mathbf{x}}$ in $\mathcal{H}$.

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 $\mathbf{x}$ in $\mathcal{X}$ will be represented by a function $\Phi(\mathbf{x}) = K_{\mathbf{x}}$ 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?

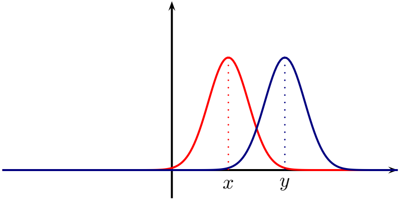
Functional spaces for machine learning

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 $\mathcal{H}$: $f(x) = \langle f, \Phi(x) \rangle$.

RKHS Definition

Definition

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

1. $\mathcal{H}$ contains all functions of the form
$$\forall \mathbf{x} \in \mathcal{X}, \quad K_{\mathbf{x}} : \mathbf{t} \mapsto K(\mathbf{x}, \mathbf{t}) \, .$$
2. For every $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}$ the reproducing property holds:
$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}} \, .$$

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

RKHS: why do we care?

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

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

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

RKHS: why do we care?

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

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

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

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 $\mathbf{x} \in \mathcal{X}$, the (linear) mapping:

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

is continuous.

An equivalent definition of RKHS

Theorem

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

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

is continuous.

Corollary

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

Proof

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

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

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

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

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

Proof (Converse)

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

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

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

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

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

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 $\mathbf{x} \in \mathcal{X}$:

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

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

Proof

If a r.k. exists then it is unique

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

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

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

The RKHS of a r.k. K is unique

Left as exercise.

An important result

Theorem

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

Proof

A r.k. is p.d.

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

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

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

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

Proof

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

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

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

let:

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

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:

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

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

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

Proof

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

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

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

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

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

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

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

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

Proof

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

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

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

  Therefore for any **x** the sequence $(f_n(\mathbf{x}))_{n\geq 0}$ is Cauchy in $\mathbb{R}$ and has therefore a limit.
- If we add to $\mathcal{H}_0$ the functions defined as the pointwise limits of Cauchy sequences, then the space becomes complete and is therefore a Hilbert space, with K as r.k. (up to a few technicalities, left as exercise). □

Application: back to Aronzsajn's theorem

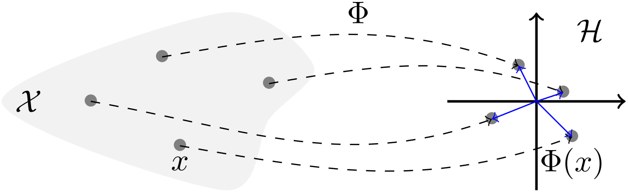
Theorem (Aronszajn, 1950)

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

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

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

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



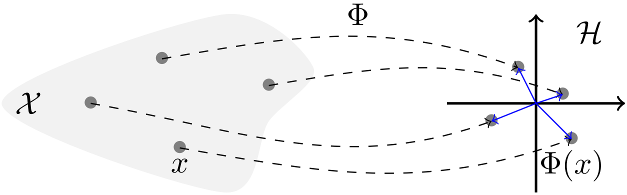
Proof of Aronzsajn's theorem

- 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:

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

- By the reproducing property we have:

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



Outline

1. Kernels and RKHS
   - 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:

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

Theorem

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

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

endowed with the inner product

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

and corresponding norm

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

Proof

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

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

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

$$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle_{\mathbb{R}^d} = \langle f_{\mathbf{w}}, K_{\mathbf{x}} \rangle_{\mathcal{H}}.$$

$\mathcal{H}$ is thus the RKHS of the linear kernel.

The polynomial kernel

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

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

The polynomial kernel

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

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

First step: Look for an inner-product.

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= \operatorname{trace}\left(\mathbf{x}^\top \mathbf{y} \; \mathbf{x}^\top \mathbf{y}\right) \\ &= \operatorname{trace}\left(\mathbf{y}^\top \mathbf{x} \; \mathbf{x}^\top \mathbf{y}\right) \\ &= \operatorname{trace}\left(\mathbf{x}\mathbf{x}^\top \mathbf{y}\mathbf{y}^\top\right) \\ &= \left\langle \mathbf{x}\mathbf{x}^\top, \mathbf{y}\mathbf{y}^\top \right\rangle_{\mathrm{F}}, \end{aligned}$$

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 $\mathcal{H}$ contains all the functions

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

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

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

where $\mathcal{S}^{d \times d}$ is the set of symmetric[1] matrices in $\mathbb{R}^{d \times d}$, endowed with the inner-product $\langle f_{\mathbf{S}_1}, f_{\mathbf{S}_2} \rangle_\mathcal{H} = \langle \mathbf{S}_1, \mathbf{S}_2 \rangle_\mathsf{F}$.

---

[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 $\mathcal{S}^{d \times d}$ by $\Phi : \mathbf{S} \mapsto f_{\mathbf{S}}$. 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 $K_{\mathbf{x}} : \mathbf{t} \mapsto K(\mathbf{x}, \mathbf{t}) = \left\langle \mathbf{x}\mathbf{x}^\top, \mathbf{t}\mathbf{t}^\top \right\rangle_{\mathrm{F}}$.
2. For all $f_{\mathbf{S}}$ in $\mathcal{H}$ and $\mathbf{x}$ in $\mathcal{X}$,

$$f_{\mathbf{S}}(\mathbf{x}) = \left\langle \mathbf{S}, \mathbf{x}\mathbf{x}^\top \right\rangle_{\mathrm{F}} = \langle f_{\mathbf{S}}, f_{\mathbf{x}\mathbf{x}^\top} \rangle_{\mathcal{H}} = \langle f_{\mathbf{S}}, K_{\mathbf{x}} \rangle_{\mathcal{H}} \,.$$

□

Remark

All points $\mathbf{x}$ in $\mathcal{X}$ are mapped to a rank-one matrix $\mathbf{x}\mathbf{x}^\top$, hence to a function $K_{\mathbf{x}} = f_{\mathbf{x}\mathbf{x}^\top}$ 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:

$$\begin{aligned} K_1 + K_2&, \\ K_1 K_2&, \text{ and} \\ cK_1&, \text{ for } c \geq 0, \end{aligned}$$

  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:

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

  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 $\mathbf{K}_1$ and $\mathbf{K}_2$. As p.s.d. matrices, they admit factorizations $\mathbf{K}_1 = \mathbf{X}^\top \mathbf{X}$ and $\mathbf{K}_2 = \mathbf{Y}^\top \mathbf{Y}$. Then,

$$\begin{aligned}
[\mathbf{K}]_{ij} &= [\mathbf{K}_1]_{ij} [\mathbf{K}_2]_{ij} \\
&= \text{trace}\left((\mathbf{x}_i^\top \mathbf{x}_j)(\mathbf{y}_j^\top \mathbf{y}_i) \right) \\
&= \text{trace}\left((\mathbf{y}_i \mathbf{x}_i^\top)(\mathbf{x}_j \mathbf{y}_j^\top) \right) \\
&= \left\langle \mathbf{x}_i \mathbf{y}_i^\top, \mathbf{x}_j \mathbf{y}_j^\top \right\rangle_{\text{F}} . \\
&= \langle \mathbf{z}_i, \mathbf{z}_j \rangle_{\mathbb{R}^{n^2}} ,
\end{aligned}$$

where the $\mathbf{x}_i$'s and the $\mathbf{y}_i$'s are the columns of $\mathbf{X}$ and $\mathbf{Y}$, respectively and $\mathbf{z}_i = \text{vec}(\mathbf{x}_i \mathbf{y}_i^\top)$. Thus, $\mathbf{K}$ is p.s.d. and $K = K_1 K_2$ is a p.d. kernel. ☐

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:

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

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?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

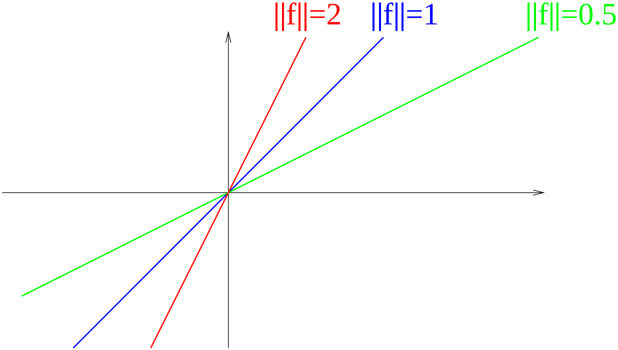
- $\mathcal{X} = (-1, 1), \quad K(x, x') = \frac{1}{1 - xx'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{x+x'}$
- $\mathcal{X} = \mathbb{N}, \quad K(x, x') = 2^{xx'}$
- $\mathcal{X} = \mathbb{R}_+, \quad K(x, x') = \log(1 + xx')$
- $\mathcal{X} = \mathbb{R}, \quad K(x, x') = \exp\left(-|x - x'|^2\right)$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

Outline

1. Kernels and RKHS
    - Positive Definite Kernels
    - Reproducing Kernel Hilbert Spaces (RKHS)
    - Examples
    - Smoothness functional
2. Kernel tricks
3. Kernel Methods: Supervised Learning
4. Kernel Methods: Unsupervised Learning
5. The Kernel Jungle
6. Open Problems and Research Topics

Remember the RKHS of the linear kernel

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



Smoothness functional

A simple inequality

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

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

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

Important message

$$\text{Small norm} \implies \text{slow variations.}$$

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.

Part 2

Kernel tricks

Motivations

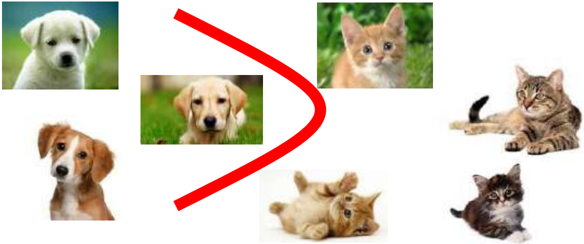
Two theoretical results underpin a family of powerful algorithms for data analysis using p.d. kernels, collectively known as kernel methods:

- 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** $f : \mathcal{X} \rightarrow \mathcal{Y}$ given labeled training data $(\mathbf{x}_i, y_i)_{i=1,\dots,n}$ with $\mathbf{x}_i$ in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

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



(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 $(\mathbf{x}_i, y_i)_{i=1,\dots,n}$ with $\mathbf{x}_i$ in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

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

The labels y_i are, for instance, in

- $\{-1, +1\}$ for **binary** classification problems.
- $\{1, \dots, K\}$ 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** $f : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(\mathbf{x}_i, y_i)_{i=1,\dots,n}$ with $\mathbf{x}_i$ in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

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

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

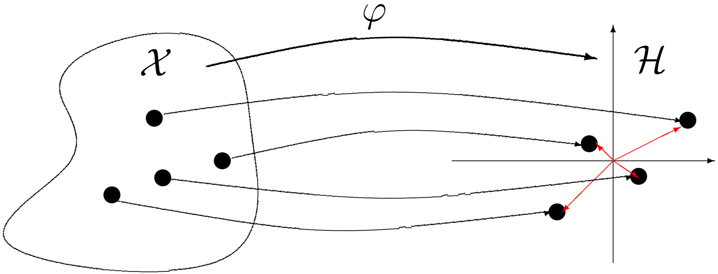
- assume there exists a linear relation between y and features $\mathbf{x}$ in $\mathbb{R}^p$.
- $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ is parametrized by $\mathbf{w}, b$ in $\mathbb{R}^{p+1}$;
- L is often a **convex** loss function;
- $\Omega(f)$ is often the squared ℓ_2-norm $\|\mathbf{w}\|^2$.

Motivation from supervised learning

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

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

$$\Phi : \mathcal{X} \to \mathcal{H} \qquad \text{and} \qquad f(\mathbf{x}) = \langle \Phi(\mathbf{x}), f\rangle_{\mathcal{H}}.$$



Motivation from supervised learning

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

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

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

First purpose: embed data in a vectorial space where

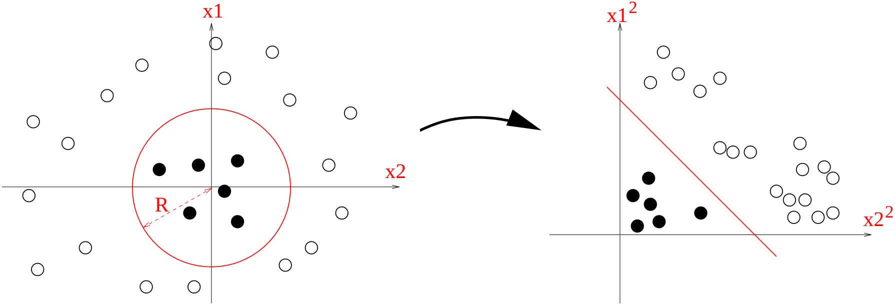
- many **geometrical operations** exist (angle computation, projection on linear subspaces, definition of barycenters....).
- one may learn potentially **rich infinite-dimensional models**.
- **regularization** is natural and theoretically grounded.

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

4. Kernel Methods: Unsupervised Learning

5. The Kernel Jungle

6. Open Problems and Research Topics

The kernel trick

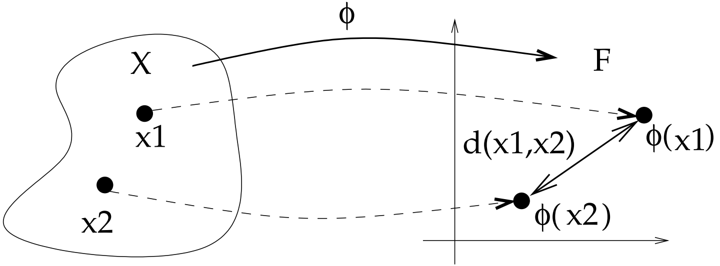
Proposition

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



$$
\begin{aligned}
d_K\left(\mathbf{x}_1,\mathbf{x}_2\right)^2 &= \|\,\Phi\left(\mathbf{x}_1\right)-\Phi\left(\mathbf{x}_2\right)\,\|_{\mathcal{H}}^2\\
&= \left\langle\Phi\left(\mathbf{x}_1\right)-\Phi\left(\mathbf{x}_2\right),\Phi\left(\mathbf{x}_1\right)-\Phi\left(\mathbf{x}_2\right)\right\rangle_{\mathcal{H}}\\
&= \left\langle\Phi\left(\mathbf{x}_1\right),\Phi\left(\mathbf{x}_1\right)\right\rangle_{\mathcal{H}}+\left\langle\Phi\left(\mathbf{x}_2\right),\Phi\left(\mathbf{x}_2\right)\right\rangle_{\mathcal{H}}-2\left\langle\Phi\left(\mathbf{x}_1\right),\Phi\left(\mathbf{x}_2\right)\right\rangle_{\mathcal{H}}
\end{aligned}
$$

$$d_K(\mathbf{x}_1,\mathbf{x}_2)^2 = K(\mathbf{x}_1,\mathbf{x}_1)+K(\mathbf{x}_2,\mathbf{x}_2)-2K(\mathbf{x}_1,\mathbf{x}_2)$$

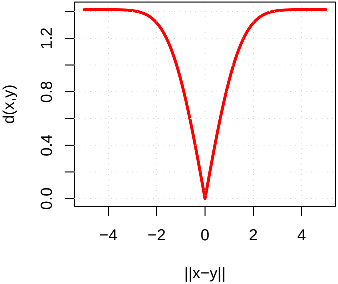
Distance for the Gaussian kernel

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

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

- $K(\mathbf{x}, \mathbf{x}) = 1 = \| \Phi(\mathbf{x}) \|_{\mathcal{H}}^2$, so all points are on the unit sphere in the feature space.
- The distance between the images of two points $\mathbf{x}$ and $\mathbf{y}$ in the feature space is given by:

$$d_K(\mathbf{x}, \mathbf{y}) = \sqrt{2\left[1 - e^{-\frac{\| \mathbf{x} - \mathbf{y} \|^2}{2\sigma^2}}\right]}$$



Example 2: distance between a point and a set

Problem

- Let $\mathcal{S} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ be a finite set of points in $\mathcal{X}$.
- How to define and compute the similarity between any point $\mathbf{x}$ in $\mathcal{X}$ and the set $\mathcal{S}$?

Example 2: distance between a point and a set

Problem

- Let $\mathcal{S} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ be a finite set of points in $\mathcal{X}$.
- How to define and compute the similarity between any point $\mathbf{x}$ in $\mathcal{X}$ and the set $\mathcal{S}$?

A solution:

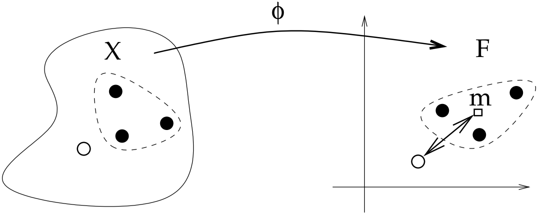
- Map all points to the feature space.
- Summarize $\mathcal{S}$ by the barycenter of the points:

$$\boldsymbol{\mu} := \frac{1}{n} \sum_{i=1}^{n} \Phi\left(\mathbf{x}_i\right) .$$

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

$$d_K\left(\mathbf{x}, \mathcal{S}\right) := \parallel \Phi\left(\mathbf{x}\right) - \boldsymbol{\mu} \parallel_{\mathcal{H}} .$$

Computation



$$d_K\left(\mathbf{x}, \mathcal{S}\right) = \left\| \Phi\left(\mathbf{x}\right) - \frac{1}{n} \sum_{i=1}^{n} \Phi(\mathbf{x}_i) \right\|_{\mathcal{H}}$$

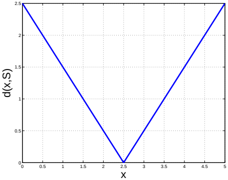
$$= \sqrt{K(\mathbf{x}, \mathbf{x}) - \frac{2}{n} \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_i) + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} K(\mathbf{x}_i, \mathbf{x}_j)}.$$

Remark

The barycentre $\boldsymbol{\mu}$ only exists in the feature space in general: it does not necessarily have a pre-image $\mathbf{x}_{\boldsymbol{\mu}}$ such that $\Phi\left(\mathbf{x}_{\boldsymbol{\mu}}\right) = \boldsymbol{\mu}$.

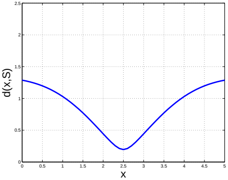
1D illustration

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



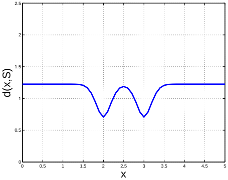
$$K(x, y) = xy.$$

(linear)



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

with $\sigma = 1$.

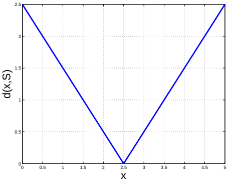


$$K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}.$$

with $\sigma = 0.2$.

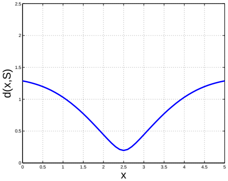
1D illustration

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



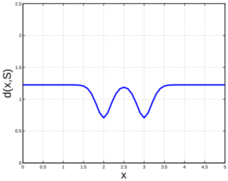
$$K(x, y) = xy.$$

(linear)



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

with $\sigma = 1$.



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

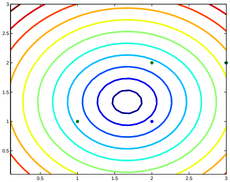
with $\sigma = 0.2$.

Remarks

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

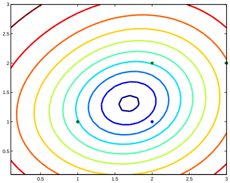
2D illustration

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



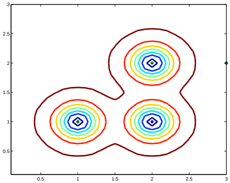
$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}\mathbf{y}.$$

(linear)



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

with $\sigma = 1$.

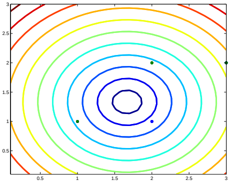


$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}}.$$

with $\sigma = 0.2$.

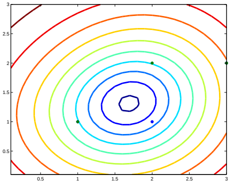
2D illustration

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



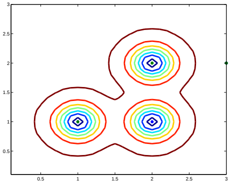
$$K(\mathbf{x}, \mathbf{y}) = \mathbf{xy}.$$

(linear)



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

with $\sigma = 1$.



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

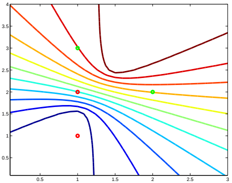
with $\sigma = 0.2$.

Remark

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

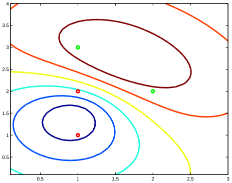
Basic application in discrimination

- $\mathcal{S}_1 = \{(1,1)', (1,2)'\}$ and $\mathcal{S}_2 = \{(1,3)', (2,2)'\}$
- Plot $f(\mathbf{x}) = d(\mathbf{x}, \mathcal{S}_1)^2 - d(\mathbf{x}, \mathcal{S}_2)^2$



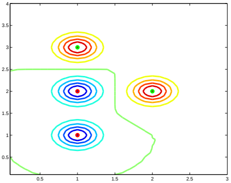
$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}\mathbf{y}.$$

(linear)



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

with $\sigma = 1$.



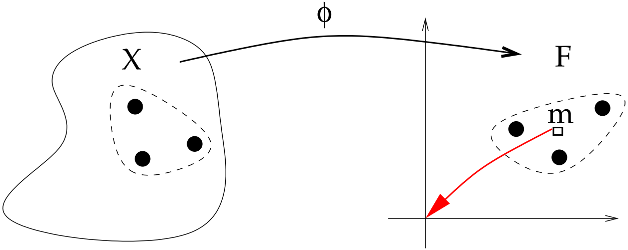
$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}}.$$

with $\sigma = 0.2$.

Example 3: Centering data in the feature space

Problem

- Let $\mathcal{S} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ be a finite set of points in $\mathcal{X}$ endowed with a p.d. kernel K. Let $\mathbf{K}$ be their $n \times n$ Gram matrix: $[\mathbf{K}]_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.
- Let $\boldsymbol{\mu} = 1/n \sum_{i=1}^{n} \Phi(\mathbf{x}_i)$ their barycenter, and $\mathbf{u}_i = \Phi(\mathbf{x}_i) - \boldsymbol{\mu}$ for $i = 1, \ldots, n$ be centered data in $\mathcal{H}$.
- How to compute the centered Gram matrix $[\mathbf{K}^c]_{i,j} = \langle \mathbf{u}_i, \mathbf{u}_j \rangle_{\mathcal{H}}$?



Computation

- A direct computation gives, for $0 \leq i, j \leq n$:

$$\begin{aligned}
\mathbf{K}^c_{i,j} &= \langle \Phi(\mathbf{x}_i) - \boldsymbol{\mu}, \Phi(\mathbf{x}_j) - \boldsymbol{\mu} \rangle_{\mathcal{H}} \\
&= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}} - \langle \boldsymbol{\mu}, \Phi(\mathbf{x}_i) + \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}} + \langle \boldsymbol{\mu}, \boldsymbol{\mu} \rangle_{\mathcal{H}} \\
&= \mathbf{K}_{i,j} - \frac{1}{n} \sum_{k=1}^{n} (\mathbf{K}_{i,k} + \mathbf{K}_{j,k}) + \frac{1}{n^2} \sum_{k,l=1}^{n} \mathbf{K}_{k,l} \, .
\end{aligned}$$

- This can be rewritten in matricial form:

$$\mathbf{K}^c = \mathbf{K} - \mathbf{U}\mathbf{K} - \mathbf{K}\mathbf{U} + \mathbf{U}\mathbf{K}\mathbf{U} = (\mathbf{I} - \mathbf{U})\, \mathbf{K}\, (\mathbf{I} - \mathbf{U}) \, ,$$

where $\mathbf{U}_{i,j} = 1/n$ 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 $\| f \|_{\mathcal{H}}$ measures the smoothness of f.
- Given a set of data $(\mathbf{x}_i \in \mathcal{X}, y_i \in \mathbb{R})_{i=1,\ldots,n}$, a natural way to estimate a regression function $f : \mathcal{X} \to \mathbb{R}$ is to solve something like:

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

  for a loss function ℓ 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 $\mathcal{S} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \subseteq \mathcal{X}$ 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:

$$\min_{f \in \mathcal{H}} \Psi \left(f\left(\mathbf{x}_1\right), \cdots, f\left(\mathbf{x}_n\right), \| f \|_{\mathcal{H}} \right) ,$$

admits a representation of the form:

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

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

$$f \in \mathrm{Span}(K_{\mathbf{x}_1}, \ldots, K_{\mathbf{x}_n}).$$

Proof (1/2)

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

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

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

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

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

Proof (2/2)

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

  $$\forall i = 1, \cdots, n, \quad f_\perp(\mathbf{x}_i) = \langle f_\perp, K_{\mathbf{x}_i} \rangle_\mathcal{H} = 0\,,$$

  because $K_{\mathbf{x}_i} = K(\mathbf{x}_i, .) \in \mathcal{H}_\mathcal{S}$ and $f_\perp \perp \mathcal{H}_\mathcal{S}$, therefore:

  $$\forall i = 1, \cdots, n, \quad f(\mathbf{x}_i) = f_\mathcal{S}(\mathbf{x}_i)\ .$$

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

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

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

□

Remarks

Often the function Ψ has the form:

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

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

- Theoretically, the minimization will enforce the norm $\| f \|_{\mathcal{H}}$ 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

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \,, \quad \text{for some } \boldsymbol{\alpha} \in \mathbb{R}^n.$$

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

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

- Furthermore,

$$\| f \|_{\mathcal{H}}^2 = \left\| \sum_{i=1}^{n} \alpha_i K_{\mathbf{x}_i} \right\|_{\mathcal{H}}^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}.$$

Practical use of the representer theorem (2/2)

- Therefore, a problem of the form

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

  is equivalent to the following n-dimensional optimization problem:

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

- This problem can usually be solved analytically or by numerical methods; we will see many examples in the next sections.

Remarks

Dual interpretations of kernel methods

Most kernel methods have two complementary interpretations:

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

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

Part 3

Kernel Methods
Supervised Learning

Supervised learning

Definition

Given:

- $\mathcal{X}$, a space of inputs,
- $\mathcal{Y}$, a space of outputs,
- $\mathcal{S}_n = (\mathbf{x}_i, y_i)_{i=1,\ldots,n}$, 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.

Supervised learning

Definition

Given:

- $\mathcal{X}$, a space of inputs,
- $\mathcal{Y}$, a space of outputs,
- $\mathcal{S}_n = (\mathbf{x}_i, y_i)_{i=1,\ldots,n}$, 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.

Depending on the nature of the output, this covers:

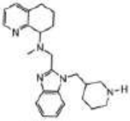
- Regression when $\mathcal{Y} = \mathbb{R}$;
- Classification when $\mathcal{Y} = \{-1, 1\}$ or any set of two labels;
- Structured output regression or classification when $\mathcal{Y}$ is more general.

Example: regression

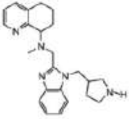
Task: predict the capacity of a small molecule to inhibit a drug target

$\mathcal{X}$ = set of molecular structures (graphs?)

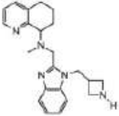
$\mathcal{Y} = \mathbb{R}$



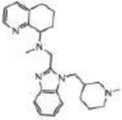
78
IC_{50}= 51±2nM



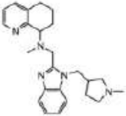
79
IC_{50}= 24±1nM



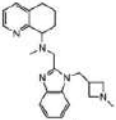
80
IC_{50}= 84±7nM



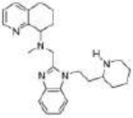
81
IC_{50}= 19±2 nM



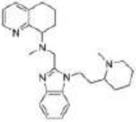
82
IC_{50}= 57±2 nM



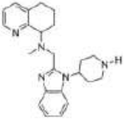
83
IC_{50}= 66±5 nM



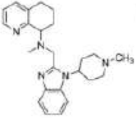
84
IC_{50}= 64±4nM



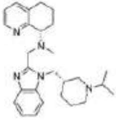
85
IC_{50}= 33±1nM



86
IC_{50}= 99±6nM



87
IC_{50}= 95±8nM



88
IC_{50}= 2.0±1nM

Example: classification

Task: recognize if an image is a dog or a cat

$\mathcal{X}$ = set of images ($\mathbb{R}^d$)

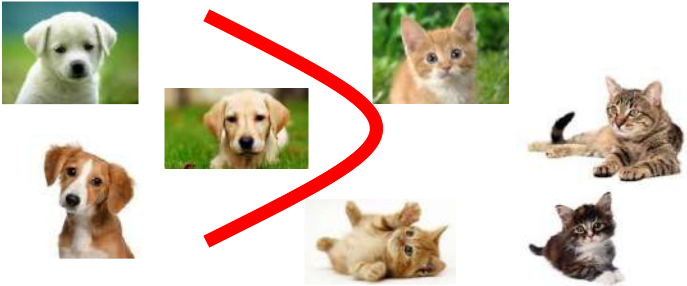
$\mathcal{Y} = \{\texttt{cat}, \texttt{dog}\}$

Example: classification

Task: recognize if an image is a dog or a cat

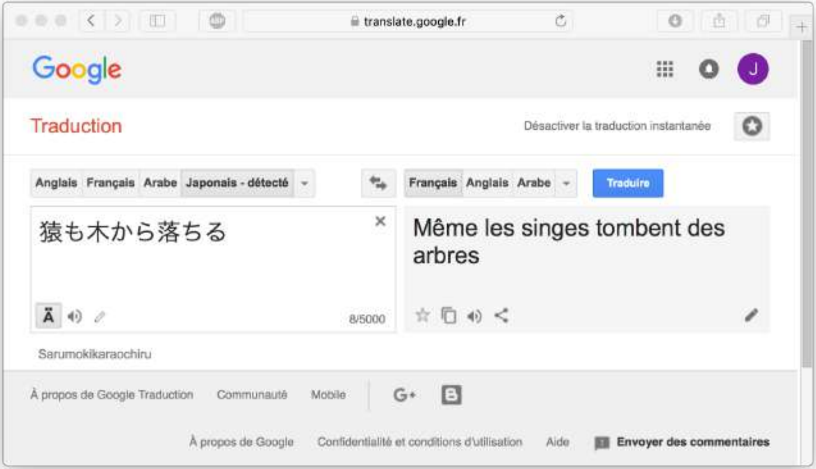
$\mathcal{X}$ = set of images ($\mathbb{R}^d$)

$\mathcal{Y} = \{\texttt{cat},\texttt{dog}\}$



Example: structured output

Task: translate from Japanese to French
$\mathcal{X}$ = finite-length strings of japanese characters
$\mathcal{Y}$ = finite-length strings of french characters



猿も木から落ちる

Sarumokikaraochiru

Même les singes tombent des arbres

Supervised learning with kernels: general principles

1. Express $h : \mathcal{X} \to \mathcal{Y}$ using a real-valued function $f : \mathcal{Z} \to \mathbb{R}$:
   - regression $\mathcal{Y} = \mathbb{R}$:
   $$h(\mathbf{x}) = f(\mathbf{x}) \quad \text{with} \quad f : \mathcal{X} \to \mathbb{R} \quad (\mathcal{Z} = \mathcal{X})$$
   - classification $\mathcal{Y} = \{-1, 1\}$:
   $$h(\mathbf{x}) = \operatorname{sign}(f(\mathbf{x})) \quad \text{with} \quad f : \mathcal{X} \to \mathbb{R} \quad (\mathcal{Z} = \mathcal{X})$$
   - structured output:
   $$h(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}, \mathbf{y}) \quad \text{with} \quad f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \quad (\mathcal{Z} = \mathcal{X} \times \mathcal{Y})$$
2. Define an empirical risk function $R_n(f)$ to assess how "good" a candidate function f is on the training set $\mathcal{S}_n$, typically the average of a loss:
$$R_n(f) := \frac{1}{n}\sum_{i=1}^{n} \ell\left(f(\mathbf{x}_i), \mathbf{y}_i\right)$$
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

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

- Regularization is important, particularly in high dimension, to prevent overfitting
- When $\mathcal{Z} = \mathbb{R}^d$ and K is the linear kernel, $f = f_{\mathbf{w}}$ is a linear model and the regularization is $\|\mathbf{w}\|^2$
- 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

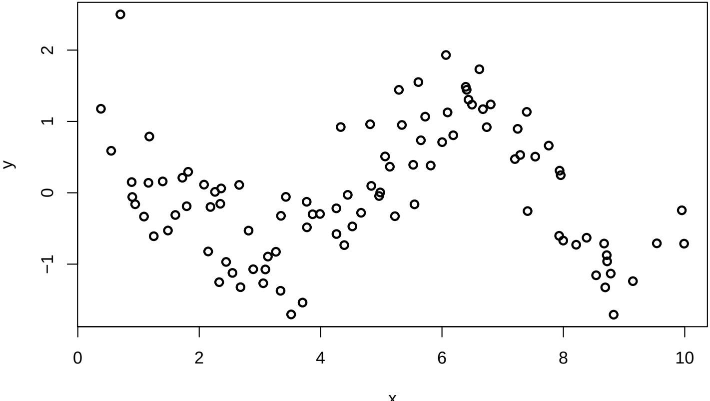
We will now see a few methods in more details

Outline

Regression

Setup

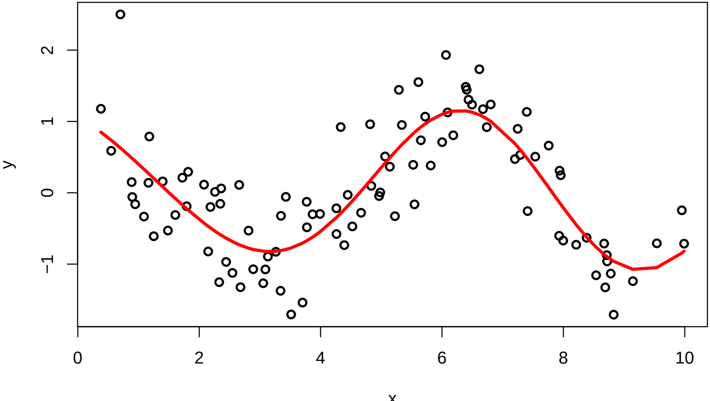
- $\mathcal{X}$ set of inputs
- $\mathcal{Y} = \mathbb{R}$ real-valued outputs
- $\mathcal{S}_n = (\mathbf{x}_i, y_i)_{i=1,\ldots,n} \in (\mathcal{X} \times \mathbb{R})^n$ a training set of n pairs
- Goal = find a function $f : \mathcal{X} \to \mathbb{R}$ to predict y by $f(\mathbf{x})$



Regression

Setup

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



Least-square regression over a general functional space

- Let us quantify the error if f predicts $f(\mathbf{x})$ instead of y by the squared error:
$$\ell(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2$$
- 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):
$$\hat{f} \in \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2$$
- 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:

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

- *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:

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

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

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

- *2nd effect = simplifying the solution.*

Solving KRR

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

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

- The following holds as usual:

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

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

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

Solving KRR

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

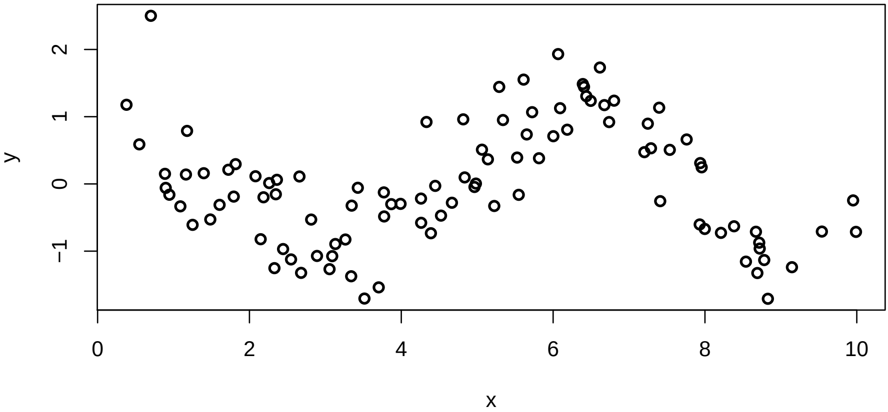
- This is a convex and differentiable function of $\boldsymbol{\alpha}$. Its minimum can therefore be found by setting the gradient in $\boldsymbol{\alpha}$ to zero:

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

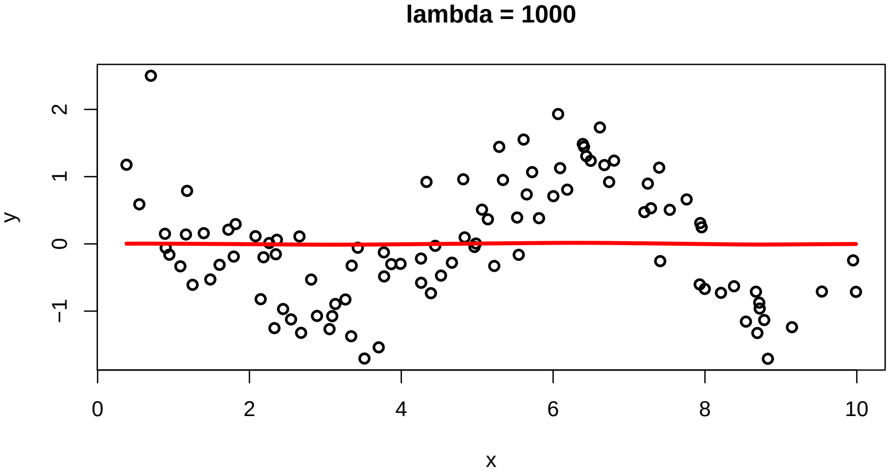
- For $\lambda > 0$, $\mathbf{K}+\lambda n\mathbf{I}$ is invertible (because $\mathbf{K}$ is positive semidefinite) so one solution is to take:

$$\boldsymbol{\alpha} = \left(\mathbf{K}+\lambda n\mathbf{I}\right)^{-1}\mathbf{y}.$$

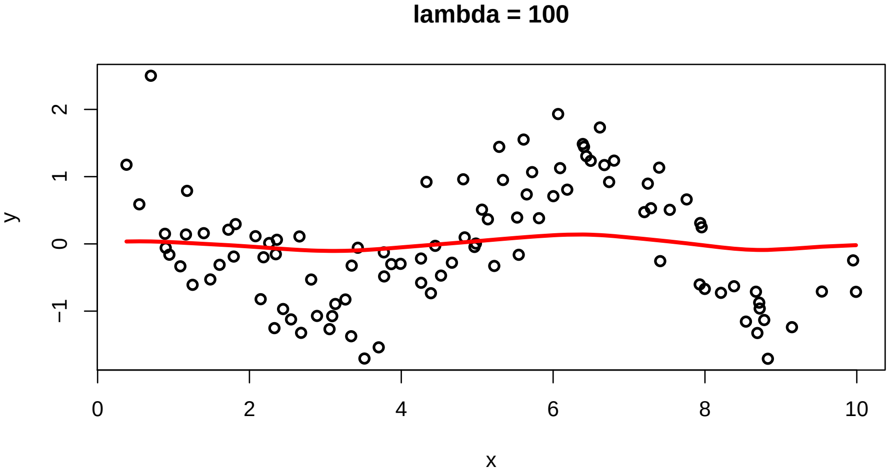
Example (KRR with Gaussian RBF kernel)



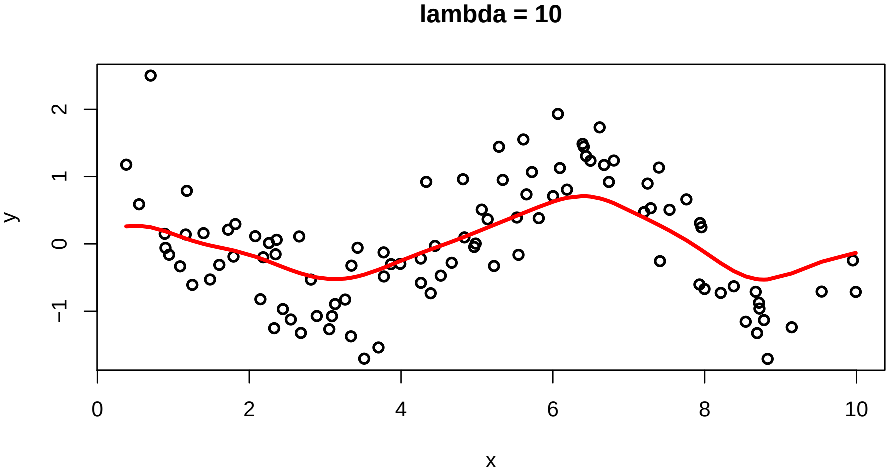
Example (KRR with Gaussian RBF kernel)



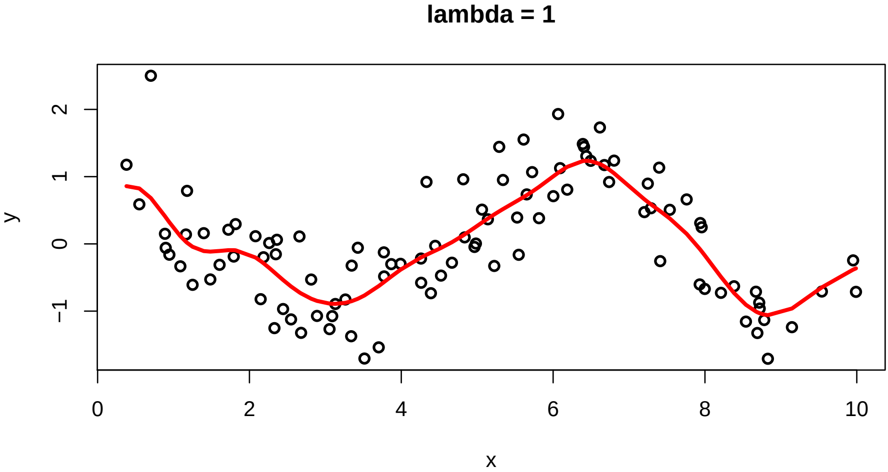
Example (KRR with Gaussian RBF kernel)



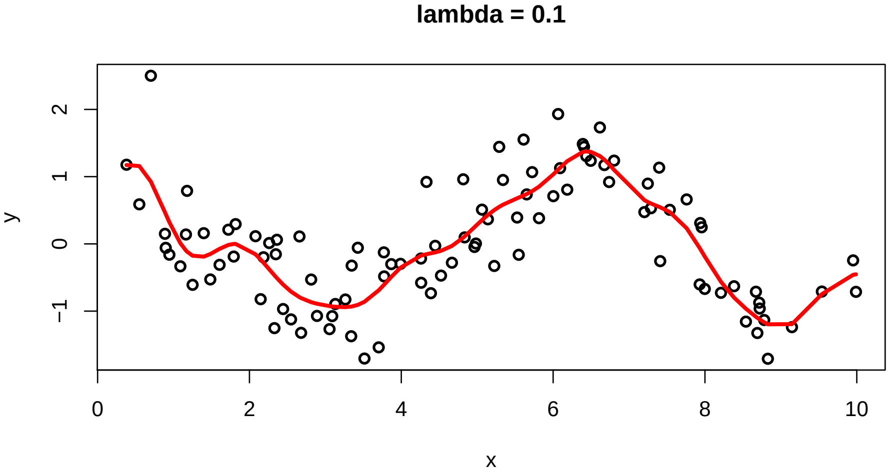
Example (KRR with Gaussian RBF kernel)



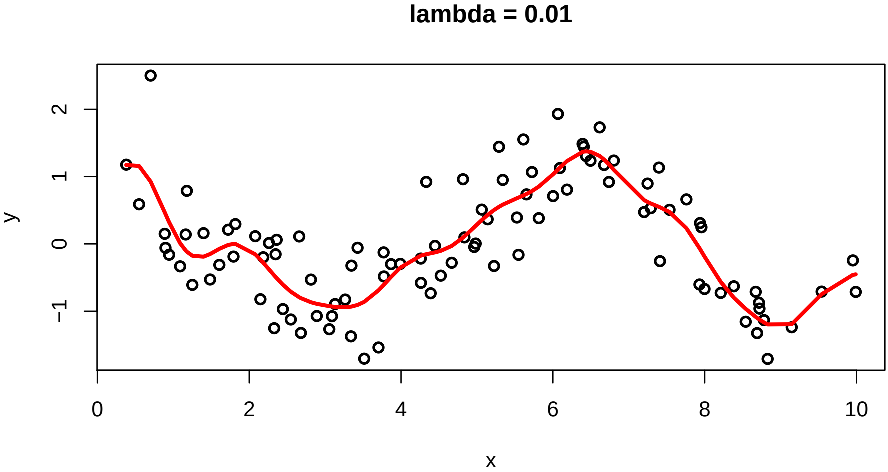
Example (KRR with Gaussian RBF kernel)



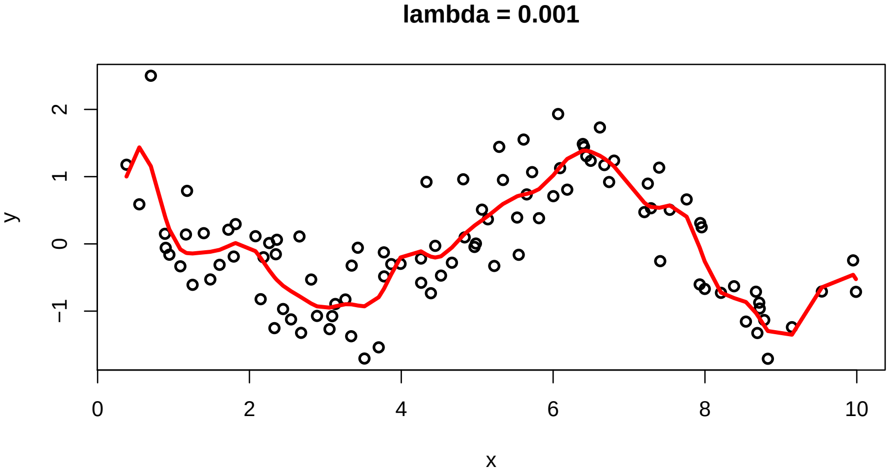
Example (KRR with Gaussian RBF kernel)



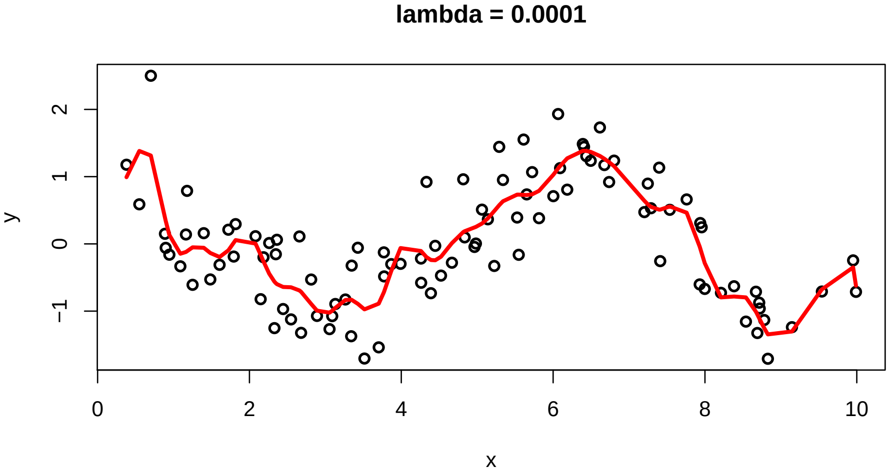
Example (KRR with Gaussian RBF kernel)



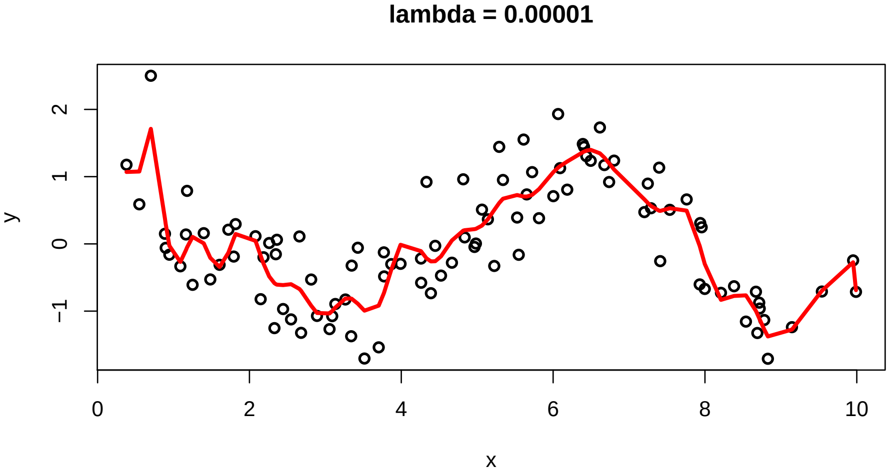
Example (KRR with Gaussian RBF kernel)



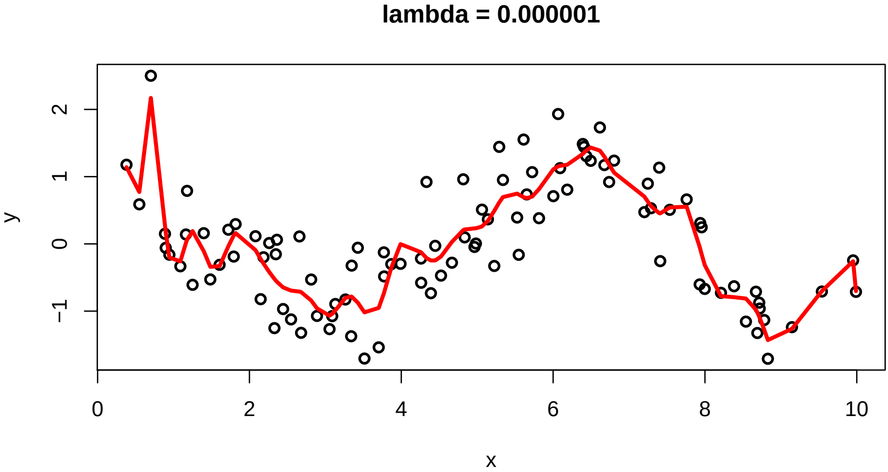
Example (KRR with Gaussian RBF kernel)



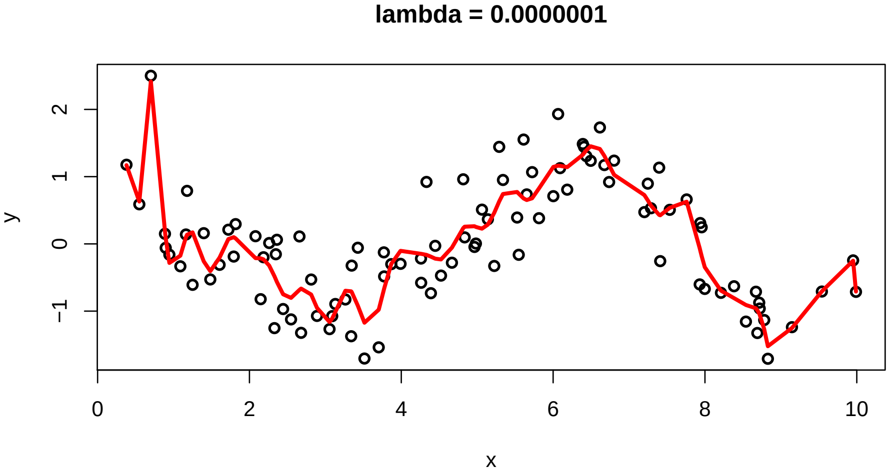
Example (KRR with Gaussian RBF kernel)



Example (KRR with Gaussian RBF kernel)



Example (KRR with Gaussian RBF kernel)



Remark: uniqueness of the solution

Let us find *all* $\boldsymbol{\alpha}$'s that solve

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

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

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

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

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

  therefore $f=f'$. KRR has a unique solution $f\in\mathcal{H}$, which can possibly be expressed by several $\boldsymbol{\alpha}$'s if K is singular.

Remark: link with "standard" ridge regression

- Take $\mathcal{X} = \mathbb{R}^d$ and the linear kernel $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top$ the $n \times d$ data matrix
- The kernel matrix is then $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$
- The function learned by KRR in that case is linear:

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

with

$$\mathbf{w}_{KRR} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i = \mathbf{X}^\top \boldsymbol{\alpha} = \mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda n \mathbf{I}\right)^{-1} \mathbf{y}$$

Remark: link with "standard" ridge regression

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

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

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

- Setting the gradient to 0 gives the solution:

$$\mathbf{w}_{RR} = \left(\mathbf{X}^\top \mathbf{X} + \lambda n \mathbf{I} \right)^{-1} \mathbf{X}^\top \mathbf{y}$$

- Oups, looks different from $\mathbf{w}_{KRR} = \mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda n \mathbf{I} \right)^{-1} \mathbf{y}$ ..?

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):

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

We deduce that (of course...):

$$\mathbf{w}_{RR} = \underbrace{\left(\mathbf{X}^\top \mathbf{X} + \lambda n \mathbf{I}\right)^{-1}}_{d \times d} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \underbrace{\left(\mathbf{X}\mathbf{X}^\top + \lambda n \mathbf{I}\right)^{-1}}_{n \times n} \mathbf{y} = \mathbf{w}_{KRR}$$

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):

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

We deduce that (of course...):

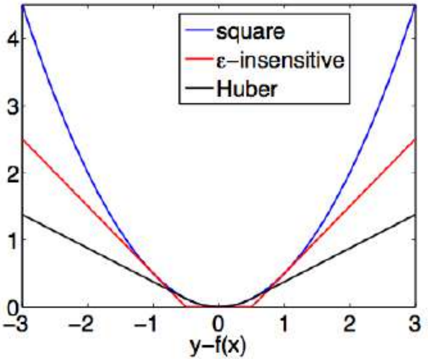
$$\mathbf{w}_{RR} = \underbrace{\left(\mathbf{X}^\top \mathbf{X} + \lambda n \mathbf{I}\right)^{-1}}_{d \times d} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \underbrace{\left(\mathbf{X}\mathbf{X}^\top + \lambda n \mathbf{I}\right)^{-1}}_{n \times n} \mathbf{y} = \mathbf{w}_{KRR}$$

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:

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

- By the representer theorem the solution is $f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K\left(\mathbf{x}_i, \mathbf{x}\right)$ where $\boldsymbol{\alpha}$ solves, with $\mathbf{W} = \mathrm{diag}\left(W_1, \ldots, W_n\right)$:

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

Weighted regression

- Setting the gradient to zero gives

$$\begin{aligned} 0 &= \frac{2}{n}\left(\mathbf{KWK}\boldsymbol{\alpha} - \mathbf{KWy}\right) + 2\lambda\mathbf{K}\boldsymbol{\alpha} \\ &= \frac{2}{n}\mathbf{KW}^{\frac{1}{2}}\left[\left(\mathbf{W}^{\frac{1}{2}}\mathbf{KW}^{\frac{1}{2}} + n\lambda\mathbf{I}\right)\mathbf{W}^{-\frac{1}{2}}\boldsymbol{\alpha} - \mathbf{W}^{\frac{1}{2}}\mathbf{y}\right] \end{aligned}$$

- A solution is therefore given by

$$\left(\mathbf{W}^{\frac{1}{2}}\mathbf{KW}^{\frac{1}{2}} + n\lambda\mathbf{I}\right)\mathbf{W}^{-\frac{1}{2}}\boldsymbol{\alpha} - \mathbf{W}^{\frac{1}{2}}\mathbf{y} = 0$$

therefore

$$\boldsymbol{\alpha} = \mathbf{W}^{\frac{1}{2}}\left(\mathbf{W}^{\frac{1}{2}}\mathbf{KW}^{\frac{1}{2}} + n\lambda\mathbf{I}\right)^{-1}\mathbf{W}^{\frac{1}{2}}\mathbf{Y}$$

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

4. Kernel Methods: Unsupervised Learning

5. The Kernel Jungle

6. Open Problems and Research Topics

Binary classification

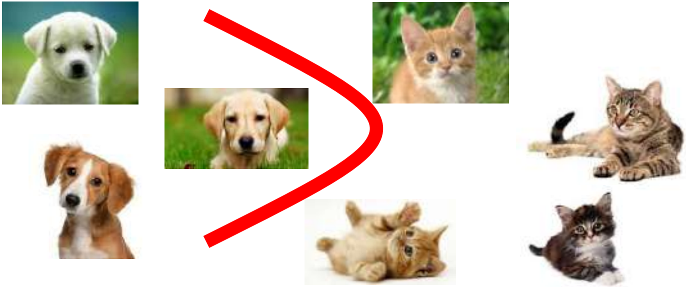
Setup

- $\mathcal{X}$ set of inputs
- $\mathcal{Y} = \{-1, 1\}$ binary outputs
- $\mathcal{S}_n = (\mathbf{x}_i, y_i)_{i=1,\dots,n} \in (\mathcal{X} \times \mathcal{Y})^n$ a training set of n pairs
- Goal = find a function $f : \mathcal{X} \to \mathbb{R}$ to predict y by $sign(f(\mathbf{x}))$

Binary classification

Setup

- $\mathcal{X}$ set of inputs
- $\mathcal{Y} = \{-1, 1\}$ binary outputs
- $\mathcal{S}_n = (\mathbf{x}_i, y_i)_{i=1,\dots,n} \in (\mathcal{X} \times \mathcal{Y})^n$ a training set of n pairs
- Goal = find a function $f : \mathcal{X} \to \mathbb{R}$ to predict y by $sign(f(\mathbf{x}))$



The 0/1 loss

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

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

- It is then tempting to learn f by solving:

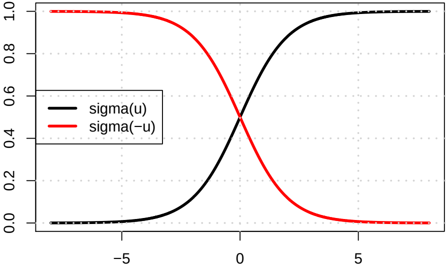
$$\underbrace{\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell_{0/1}\left(f\left(\mathbf{x}_i\right), y_i\right)}_{\text{misclassification rate}} + \underbrace{\lambda \| f \|_{\mathcal{H}}^2}_{\text{regularization}}$$

- 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 $f(\mathbf{x})$, e.g.:

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



- The logistic loss is the negative conditional likelihood:

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

Kernel logistic regression (KLR)

$$\begin{aligned}\hat{f} &= \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell_{logistic}\left(f(\mathbf{x}_i), y_i\right) + \frac{\lambda}{2} \| f \|_{\mathcal{H}}^2 \\ &= \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(1 + e^{-y_i f(\mathbf{x}_i)}\right) + \frac{\lambda}{2} \| f \|_{\mathcal{H}}^2\end{aligned}$$

- 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

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

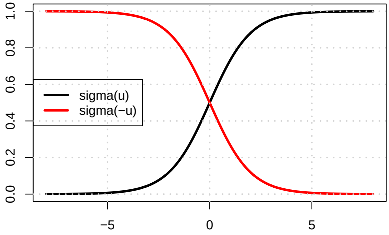
  and as always we have:

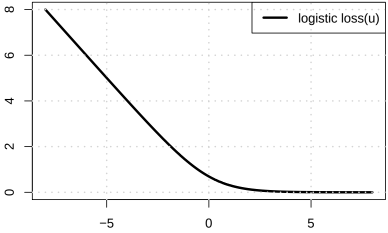
$$\left(\hat{f}\left(\mathbf{x}_1\right), \ldots, \hat{f}\left(\mathbf{x}_n\right)\right)^\top = \mathbf{K}\boldsymbol{\alpha} \quad \text{and} \quad \| \hat{f} \|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$$

- To find $\boldsymbol{\alpha}$ we therefore need to solve:

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

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:

- $\ell_{logistic}(u) = \ln\left(1 + e^{-u}\right)$
- $\ell'_{logistic}(u) = -\sigma(-u)$
- $\ell''_{logistic}(u) = \sigma(u)\sigma(-u) \geq 0$

Back to KLR

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} J(\boldsymbol{\alpha}) = \frac{1}{n} \sum_{i=1}^{n} \ell_{logistic}\left(y_i [\mathbf{K}\boldsymbol{\alpha}]_i\right) + \frac{\lambda}{2} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$$

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 $\boldsymbol{\alpha}_0$ is the function:

$$J_q(\boldsymbol{\alpha}) = J(\boldsymbol{\alpha}_0) + (\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)^\top \nabla J(\boldsymbol{\alpha}_0) + \frac{1}{2} (\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)^\top \nabla^2 J(\boldsymbol{\alpha}_0) (\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)$$

Let us compute the different terms...

Computing the quadratic approximation

$$\frac{\partial J}{\partial \alpha_j} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell'_{logistic}\left(y_i[\mathbf{K}\boldsymbol{\alpha}]_i\right)}_{P_i(\boldsymbol{\alpha})} y_i \mathbf{K}_{ij} + \lambda[\mathbf{K}\boldsymbol{\alpha}]_j$$

therefore

$$\nabla J\left(\boldsymbol{\alpha}\right) = \frac{1}{n}\mathbf{K}\mathbf{P}\left(\boldsymbol{\alpha}\right)\mathbf{y} + \lambda\mathbf{K}\boldsymbol{\alpha}$$

where $\mathbf{P}\left(\boldsymbol{\alpha}\right) = \mathrm{diag}\left(P_1(\boldsymbol{\alpha}), \ldots, P_n(\boldsymbol{\alpha})\right)$.

$$\frac{\partial^2 J}{\partial \alpha_j \partial \alpha_l} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell''_{logistic}\left(y_i[\mathbf{K}\boldsymbol{\alpha}]_i\right)}_{W_i(\boldsymbol{\alpha})} y_i \mathbf{K}_{ij} y_i \mathbf{K}_{il} + \lambda\mathbf{K}_{jl}$$

therefore

$$\nabla^2 J\left(\boldsymbol{\alpha}\right) = \frac{1}{n}\mathbf{K}\mathbf{W}\left(\boldsymbol{\alpha}\right)\mathbf{K} + \lambda\mathbf{K}$$

where $\mathbf{W}\left(\boldsymbol{\alpha}\right) = \mathrm{diag}\left(W_1(\boldsymbol{\alpha}), \ldots, W_n(\boldsymbol{\alpha})\right)$.

Computing the quadratic approximation

$$J_q(\boldsymbol{\alpha}) = J(\boldsymbol{\alpha}_0) + (\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)^\top \nabla J(\boldsymbol{\alpha}_0) + \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)^\top \nabla^2 J(\boldsymbol{\alpha}_0)(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)$$

Terms that depend on $\boldsymbol{\alpha}$, with $\mathbf{P} = \mathbf{P}(\boldsymbol{\alpha}_0)$ and $\mathbf{W} = \mathbf{W}(\boldsymbol{\alpha}_0)$:

- $\boldsymbol{\alpha}^\top \nabla J(\boldsymbol{\alpha}_0) = \frac{1}{n}\boldsymbol{\alpha}^\top \mathbf{KPy} + \lambda \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}_0$
- $\frac{1}{2}\boldsymbol{\alpha}^\top \nabla^2 J(\boldsymbol{\alpha}_0)\boldsymbol{\alpha} = \frac{1}{2n}\boldsymbol{\alpha}^\top \mathbf{KWK}\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$
- $-\boldsymbol{\alpha}^\top \nabla^2 J(\boldsymbol{\alpha}_0)\boldsymbol{\alpha}_0 = -\frac{1}{n}\boldsymbol{\alpha}^\top \mathbf{KWK}\boldsymbol{\alpha}_0 - \lambda \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}_0$

Putting it all together:

$$\begin{aligned} 2J_q(\boldsymbol{\alpha}) &= -\frac{2}{n}\boldsymbol{\alpha}^\top \mathbf{KW}\underbrace{\left(\mathbf{K}\boldsymbol{\alpha}_0 - \mathbf{W}^{-1}\mathbf{Py}\right)}_{:=\mathbf{z}} + \frac{1}{n}\boldsymbol{\alpha}^\top \mathbf{KWK}\boldsymbol{\alpha} + \lambda \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} + C \\ &= \frac{1}{n}(\mathbf{K}\boldsymbol{\alpha} - \mathbf{z})^\top \mathbf{W}(\mathbf{K}\boldsymbol{\alpha} - \mathbf{z}) + \lambda \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} + C \end{aligned}$$

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:

$$\boldsymbol{\alpha}^{t+1} \leftarrow \mathrm{solveWKRR}(\mathbf{K}, \mathbf{W}^t, \mathbf{z}^t)$$

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

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

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
4. Kernel Methods: Unsupervised Learning
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 $\ell_{0/1}\left(f(\mathbf{x}), y\right) = \mathbf{1}\left(yf(\mathbf{x}) < 0\right)$
- The logistic loss $\ell_{logistic}\left(f(\mathbf{x}), y\right) = \ln\left(1 + e^{-yf(\mathbf{x})}\right)$

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

Definition

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

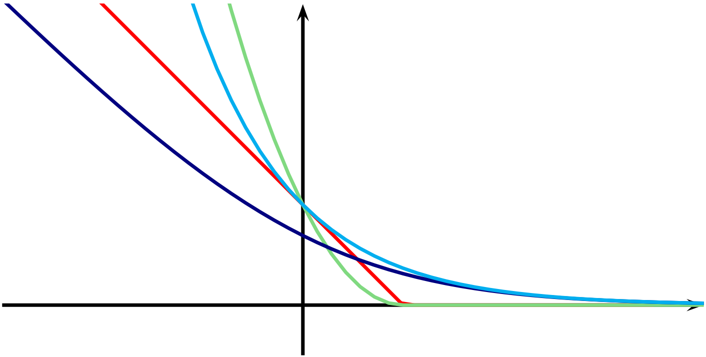
$$yf\left(\mathbf{x}\right).$$

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

$$\ell\left(f(\mathbf{x}), y\right) = \varphi\left(yf(\mathbf{x})\right), \quad \text{with } \varphi \text{ non-increasing}$$

What about other similar loss functions?

Loss function examples



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

Large-margin classifiers

Definition

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

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

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

Questions:

1. Can we solve the optimization problem for other φ'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

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

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

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

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

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

- When φ 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 $\eta(\mathbf{x}) = \mathbb{P}(Y = 1 \,|\, X = \mathbf{x})$ a measurable version of the conditional distribution of Y given X
- Assume the training set $\mathcal{S}_n = (X_i, Y_i)_{i=1,\dots,n}$ are i.i.d. random variables according to $\mathbb{P}$.
- The risk of a classifier $f : \mathcal{X} \to \mathbb{R}$ is $R(f) = \mathbb{P}\left(sign(f(X)) \neq Y\right)$
- The Bayes risk is
$$R^* = \inf_{f \text{ measurable}} R(f)$$
which is attained for $f^*(\mathbf{x}) = \eta(\mathbf{x}) - 1/2$
- The empirical risk of a classifier $f : \mathcal{X} \to \mathbb{R}$ is
$$R^n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left(sign(f(X_i)) \neq Y_i\right)$$

φ-risk

- Let the empirical φ-risk be the empirical risk optimized by a large-margin classifier:

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

- It is the empirical version of the φ-risk

$$R_\varphi(f) = \mathbb{E}[\varphi\left(Yf\left(X\right)\right)]$$

- Can we hope to have a small risk $R(f)$ if we focus instead on the φ-risk $R_\varphi(f)$?

A small φ-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 φ-risk is a good idea even if our focus is on the classification error.
- The assumptions on φ 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 φ-risk ensures a small $0/1$ risk

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

Condition on $X = \mathbf{x}$:

$$R_\varphi(f \,|\, X = \mathbf{x}) = \mathbb{E}\left[\varphi\left(Yf\left(X\right)\right) \,|\, X = \mathbf{x}\right] = \eta(\mathbf{x})\varphi\left(f(\mathbf{x})\right) + \left(1 - \eta(\mathbf{x})\right)\varphi\left(-f(\mathbf{x})\right)$$

$$R_\varphi(-f \,|\, X = \mathbf{x}) = \mathbb{E}\left[\varphi\left(-Yf\left(X\right)\right) \,|\, X = \mathbf{x}\right] = \eta(\mathbf{x})\varphi\left(-f(\mathbf{x})\right) + \left(1 - \eta(\mathbf{x})\right)\varphi\left(f(\mathbf{x})\right)$$

Therefore:

$$R_\varphi(f \,|\, X = \mathbf{x}) - R_\varphi(-f \,|\, X = \mathbf{x}) = \left[2\eta(\mathbf{x}) - 1\right] \times \left[\varphi\left(f(\mathbf{x})\right) - \varphi\left(-f(\mathbf{x})\right)\right]$$

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

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

These inequalities are in fact strict thanks to the assumptions we made on φ (*left as exercice*). □

Empirical risk minimization (ERM)

To find a function with a small φ-risk, the following is a good candidate:

Definition

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

$$\hat{f}_n = \operatorname*{argmin}_{f \in \mathcal{F}} R_\varphi^n(f)\,.$$

Empirical risk minimization (ERM)

To find a function with a small φ-risk, the following is a good candidate:

Definition

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

$$\hat{f}_n = \operatorname*{argmin}_{f \in \mathcal{F}} R_\varphi^n(f) \,.$$

Questions

- Is $R_\varphi^n(f)$ 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 φ-risk, the following is a good candidate:

Definition

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

$$\hat{f}_n = \operatorname*{argmin}_{f \in \mathcal{F}} R_\varphi^n(f)\,.$$

Questions

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

$$R_\varphi(\hat{f}_n) - R_\varphi^\star = \underbrace{R_\varphi(\hat{f}_n) - \inf_{f \in \mathcal{F}} R_\varphi(f)}_{\text{estimation error}} + \underbrace{\inf_{f \in \mathcal{F}} R_\varphi(f) - R_\varphi^\star}_{\text{approximation error}}\,.$$

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:

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

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 φ is Lipschitz with constant L_φ:

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

Then the φ-risk of the ERM estimator satisfies (on average over the sampling of training set)

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

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

Theorem (capacity control of RKHS balls)

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

Proof (1/2)

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

Proof (2/2)

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

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

Basic learning bounds in RKHS balls

Corollary

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

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

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 λ, next slide) is tuned by cross-validation

ERM as penalized risk minimization

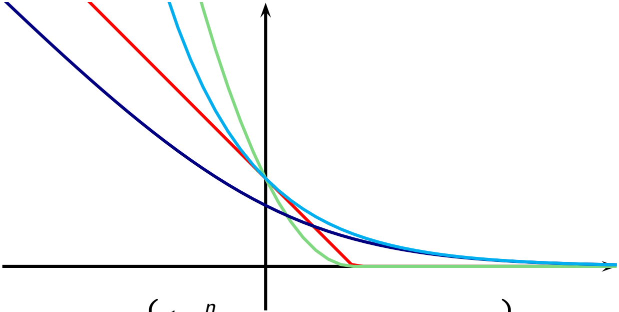
- ERM over $\mathcal{F}_B$ solves the constrained minimization problem:

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

- To make this practical we assume that φ 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 λ the unconstrained problem:

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

Summary: large margin classifiers



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

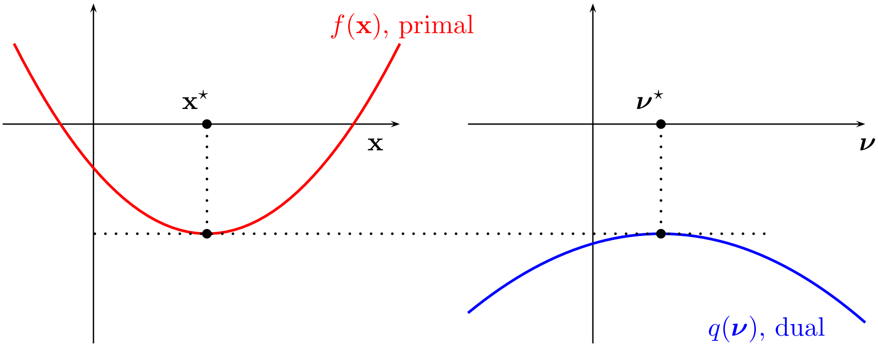
- φ calibrated (e.g., decreasing, $\varphi'(0) < 0$) $\implies$ good proxy for classification error
- φ convex + representer theorem $\implies$ efficient algorithms

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

A few slides on convex duality

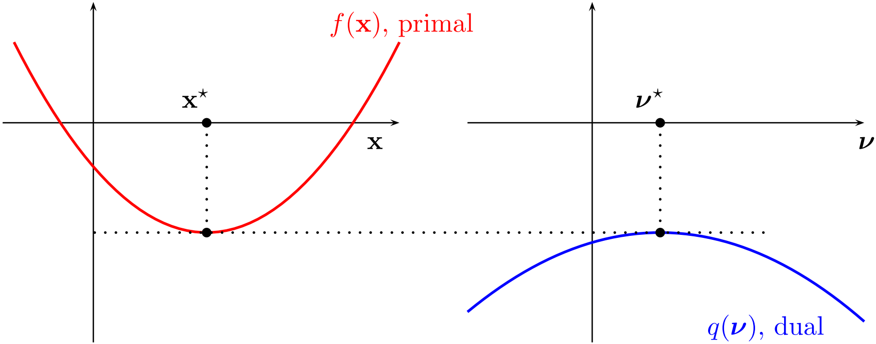
Strong Duality



- Strong duality means that $\max_{\boldsymbol{\nu}} q(\boldsymbol{\nu}) = \min_{\mathbf{x}} f(\mathbf{x})$
- Strong duality holds in most “reasonable cases” for convex optimization (to be detailed soon).

A few slides on convex duality

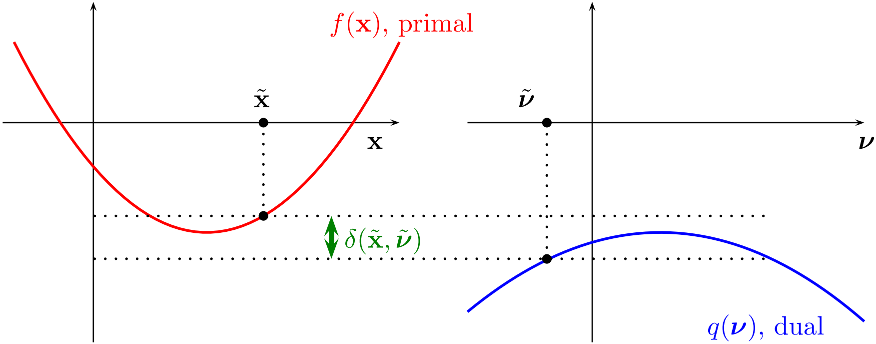
Strong Duality



- The relation between $\mathbf{x}^\star$ and $\boldsymbol{\nu}^\star$ is not always known a priori.

A few slides on convex duality

Parenthesis on duality gaps



- The duality gap guarantees us that $0 \leq f(\tilde{\mathbf{x}}) - f(\mathbf{x}^\star) \leq \delta(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\nu}})$.
- 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 $\mathbf{x} \in \mathcal{X}$:

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

  making no assumption of f, g and h.
- Let us denote by $f^\star$ the optimal value of the decision function under the constraints, i.e., $f^\star = f\left(\mathbf{x}^\star\right)$ if the minimum is reached at a global minimum $\mathbf{x}^\star$.

A few slides on Lagrangian duality

Lagrangian

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

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

Lagrangian dual function

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

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

A few slides on convex Lagrangian duality

For the (primal) problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h(\mathbf{x}) = 0 \ , \quad g(\mathbf{x}) \le 0 \ , \end{aligned}$$

the Lagrange dual problem is:

$$\begin{aligned} \text{maximize} \quad & q(\boldsymbol{\lambda}, \boldsymbol{\mu}) \\ \text{subject to} \quad & \boldsymbol{\mu} \ge 0 \ , \end{aligned}$$

Proposition

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

$$q(\boldsymbol{\lambda}, \boldsymbol{\mu}) \le f^\star \ , \quad \forall \boldsymbol{\lambda} \in \mathbb{R}^m, \forall \boldsymbol{\mu} \in \mathbb{R}^r, \boldsymbol{\mu} \ge 0 \ .$$

Proofs

- Remember that

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

- For each $\mathbf{x}$, the function $(\boldsymbol{\lambda}, \boldsymbol{\mu}) \mapsto L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is linear, and therefore both convex and concave in $(\boldsymbol{\lambda}, \boldsymbol{\mu})$. The pointwise minimum of concave functions is concave, therefore q is concave.
- Let $\bar{\mathbf{x}}$ be any feasible point, i.e., $h(\bar{\mathbf{x}}) = 0$ and $g(\bar{\mathbf{x}}) \leq 0$. Then we have, for any $\boldsymbol{\lambda}$ and $\boldsymbol{\mu} \geq 0$:

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

$$\Longrightarrow \quad L(\bar{\mathbf{x}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\bar{\mathbf{x}}) + \sum_{i=1}^{m} \lambda_i h_i(\bar{\mathbf{x}}) + \sum_{i=1}^{r} \mu_i g_i(\bar{\mathbf{x}}) \leq f(\bar{\mathbf{x}}) ,$$

$$\Longrightarrow \quad q(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq L(\bar{\mathbf{x}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f(\bar{\mathbf{x}}) , \quad \forall \bar{\mathbf{x}} . \quad \square$$

Weak duality

- Let q^* the optimal value of the Lagrange dual problem. Each $q(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is a lower bound for $f^\star$ and by definition $q^\star$ is the best lower bound that is obtained. The following weak duality inequality therefore always hold:

$$q^\star \leq f^\star \ .$$

- This inequality holds when $q^\star$ or $f^\star$ are infinite. The difference $q^\star - f^\star$ is called the optimal duality gap of the original problem.

Strong duality

- We say that strong duality holds if the optimal duality gap is zero, i.e.:

$$q^\star = f^\star \ .$$

- 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 $\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}$,

$$q(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq q(\boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = L\left(\mathbf{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star\right) = f\left(\mathbf{x}^\star\right) \leq f\left(\mathbf{x}\right).$$

Slater's constraint qualification

Strong duality holds for a convex problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0 \ , \quad j = 1, \ldots, r \ , \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \ , \end{aligned}$$

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

$$g_j(\mathbf{x}) < 0 \ , \quad j = 1, \ldots, r \ , \quad \mathbf{A}\mathbf{x} = \mathbf{b} \ .$$

Remarks

- Slater's conditions also ensure that the maximum $q^\star$ (if $> -\infty$) is attained, i.e., there exists a point $(\boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ with

$$q\left(\boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star\right) = q^\star = f^\star$$

- 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, $\mathbf{x}^\star$ is primal optimal, $(\boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ is dual optimal. Then we have:

$$
\begin{aligned}
f(\mathbf{x}^\star) &= q\left(\boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star\right) \\
&= \inf_{\mathbf{x}\in\mathbb{R}^n} \left\{ f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i^\star h_i(\mathbf{x}) + \sum_{j=1}^{r} \mu_j^\star g_j(\mathbf{x}) \right\} \\
&\leq f(\mathbf{x}^\star) + \sum_{i=1}^{m} \lambda_i^\star h_i(\mathbf{x}^\star) + \sum_{j=1}^{r} \mu_j^\star g_j(\mathbf{x}^\star) \\
&\leq f(\mathbf{x}^\star)
\end{aligned}
$$

Hence both inequalities are in fact equalities.

Complimentary slackness

The first equality shows that:

$$L\left(\mathbf{x}^{\star}, \boldsymbol{\lambda}^{\star}, \boldsymbol{\mu}^{\star}\right) = \inf_{\mathbf{x} \in \mathbb{R}^n} L\left(\mathbf{x}, \boldsymbol{\lambda}^{\star}, \boldsymbol{\mu}^{\star}\right) \ ,$$

showing that $\mathbf{x}^{\star}$ minimizes the Lagrangian at $(\boldsymbol{\lambda}^{\star}, \boldsymbol{\mu}^{\star})$. 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:

$$\mu_j g_j(\mathbf{x}^{\star}) = 0 \ , \quad j = 1, \ldots, r \ .$$

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)

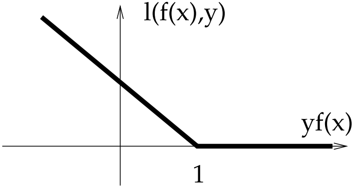
Definition

- 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:

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



Problem reformulation (1/3)

- By the representer theorem, the solution satisfies

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

  where $\hat{\boldsymbol{\alpha}}$ solves

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

- This is a convex optimization problem
- But the objective function is not smooth (because of the hinge loss)

Problem reformulation (2/3)

- Let us introduce additional slack variables $\xi_1, \ldots, \xi_n \in \mathbb{R}$. The problem is equivalent to:

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

subject to:

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

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

$$u \geq \varphi_{\text{hinge}}(v) \quad \Leftrightarrow \begin{cases} u & \geq 1 - v \\ u & \geq 0 \end{cases}$$

Problem reformulation (3/3)

In summary, the SVM solution is

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

where $\hat{\boldsymbol{\alpha}}$ solves:

SVM (primal formulation)

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

subject to:

$$\begin{cases} y_i[\mathbf{K}\boldsymbol{\alpha}]_i + \xi_i - 1 \geq 0 \,, & \text{for } i = 1, \ldots, n \,, \\ \xi_i \geq 0 \,, & \text{for } i = 1, \ldots, n \,. \end{cases}$$

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).

Lagrangian

- Let us introduce the Lagrange multipliers $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{\nu} \in \mathbb{R}^n$.
- The Lagrangian of the problem is:

$$L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\nu}) = \frac{1}{n}\sum_{i=1}^{n} \xi_i + \lambda \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} - \sum_{i=1}^{n} \mu_i \left[y_i [\mathbf{K}\boldsymbol{\alpha}]_i + \xi_i - 1 \right] - \sum_{i=1}^{n} \nu_i \xi_i$$

or, in matrix notations:

$$L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\nu}) = \boldsymbol{\xi}^\top \frac{\mathbf{1}}{n} + \lambda \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} - (\operatorname{diag}(\mathbf{y})\boldsymbol{\mu})^\top \mathbf{K}\boldsymbol{\alpha} - (\boldsymbol{\mu} + \boldsymbol{\nu})^\top \boldsymbol{\xi} + \boldsymbol{\mu}^\top \mathbf{1}$$

Minimizing $L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\nu})$ w.r.t. $\boldsymbol{\alpha}$

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

$$\nabla_{\boldsymbol{\alpha}} L = 2\lambda \mathbf{K}\boldsymbol{\alpha} - \mathbf{K}\,\mathrm{diag}\,(\mathbf{y})\boldsymbol{\mu} = \mathbf{K}\,(2\lambda\boldsymbol{\alpha} - \mathrm{diag}\,(\mathbf{y})\boldsymbol{\mu})$$

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

$$\boldsymbol{\alpha}^{*} = \frac{\mathrm{diag}\,(\mathbf{y})\boldsymbol{\mu}}{2\lambda}$$

Minimizing $L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\nu})$ w.r.t. $\boldsymbol{\xi}$

- $L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\nu})$ is a linear function in $\boldsymbol{\xi}$.
- Its minimum is $-\infty$ except when it is constant, i.e., when:

$$\nabla_{\boldsymbol{\xi}} L = \frac{\mathbf{1}}{n} - \boldsymbol{\mu} - \boldsymbol{\nu} = 0$$

or equivalently

$$\boldsymbol{\mu} + \boldsymbol{\nu} = \frac{\mathbf{1}}{n}$$

Dual function

- We therefore obtain the Lagrange dual function:

$$q\left(\boldsymbol{\mu},\boldsymbol{\nu}\right)=\inf_{\boldsymbol{\alpha}\in\mathbb{R}^n,\boldsymbol{\xi}\in\mathbb{R}^n} L\left(\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\mu},\boldsymbol{\nu}\right)$$

$$=\begin{cases}\boldsymbol{\mu}^\top\mathbf{1}-\frac{1}{4\lambda}\boldsymbol{\mu}^\top\operatorname{diag}\left(\mathbf{y}\right)\mathbf{K}\operatorname{diag}\left(\mathbf{y}\right)\boldsymbol{\mu} & \text{if } \boldsymbol{\mu}+\boldsymbol{\nu}=\frac{\mathbf{1}}{n}\,,\\ -\infty & \text{otherwise.}\end{cases}$$

- The dual problem is:

$$\begin{aligned}\text{maximize}\quad & q\left(\boldsymbol{\mu},\boldsymbol{\nu}\right)\\ \text{subject to}\quad & \boldsymbol{\mu}\geq 0\,,\boldsymbol{\nu}\geq 0\,.\end{aligned}$$

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 $q(\boldsymbol{\mu}, \boldsymbol{\nu}) = -\infty$.
- If $0 \leq \mu_i \leq 1/n$ for all i, then the dual function takes finite values that depend only on $\boldsymbol{\mu}$ by taking $\nu_i = 1/n - \mu_i$.
- The dual problem is therefore equivalent to:

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

or with indices:

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

Back to the primal

- Once the dual problem is solved in $\boldsymbol{\mu}$ we get a solution of the primal problem by $\boldsymbol{\alpha} = \mathrm{diag}(\mathbf{y})\boldsymbol{\mu}/2\lambda$.
- Because the link is so simple, we can therefore directly plug this into the dual problem to obtain the QP that $\boldsymbol{\alpha}$ must solve:

SVM (dual formulation)

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

subject to:

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

Complimentary slackness conditions

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

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

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

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

Analysis of KKT conditions

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

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

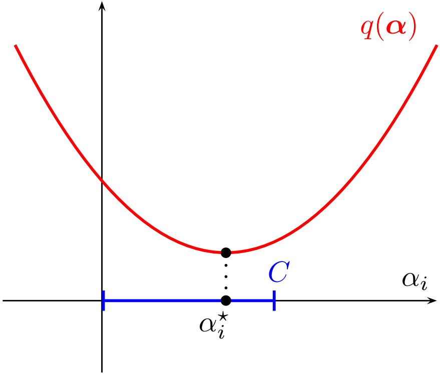
Another point of view without KKT

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

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

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

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



- Case 1: $0 < y_i\boldsymbol{\alpha}_i^\star < C$;
- $[\nabla q(\boldsymbol{\alpha}^\star)]_i = 0$;
- $\Rightarrow y_i f(\mathbf{x}_i) = 1$.

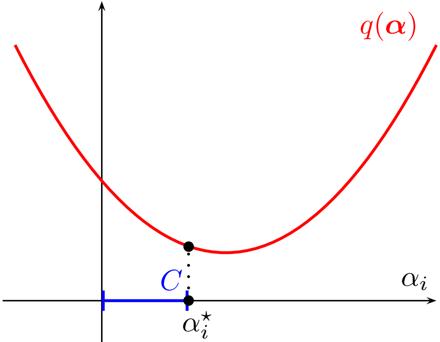
Another point of view without KKT

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

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

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

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



- Case 2: $y_i\boldsymbol{\alpha}_i^\star = C$;
- $[\nabla q(\boldsymbol{\alpha}^\star)]_i \leq 0$;
- $\Rightarrow y_i f(\mathbf{x}_i) \leq 1$.

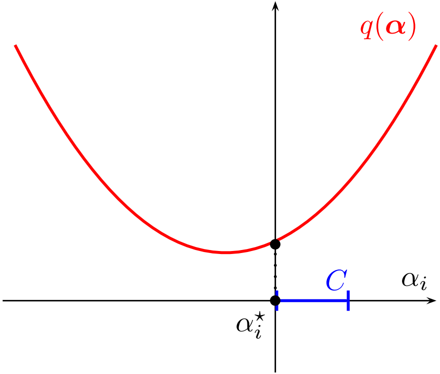
Another point of view without KKT

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

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

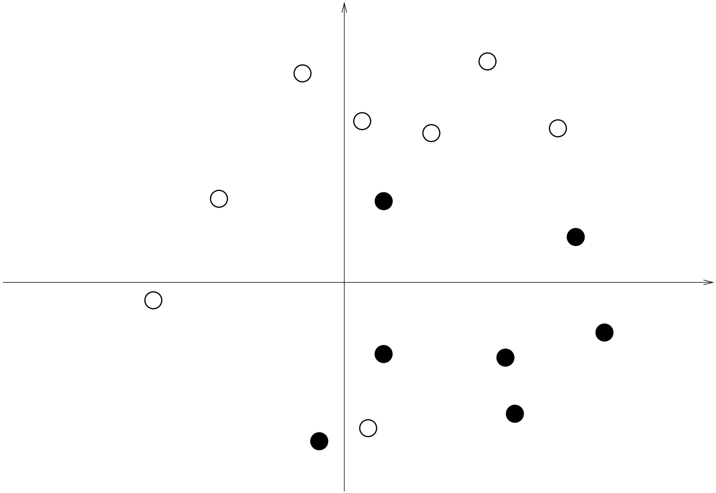
The gradient is $\nabla q(\boldsymbol{\alpha}) = \mathbf{K}\boldsymbol{\alpha} - \mathbf{y} = [f(\mathbf{x}_i) - y_i]_{i=1,\ldots,n}$.

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

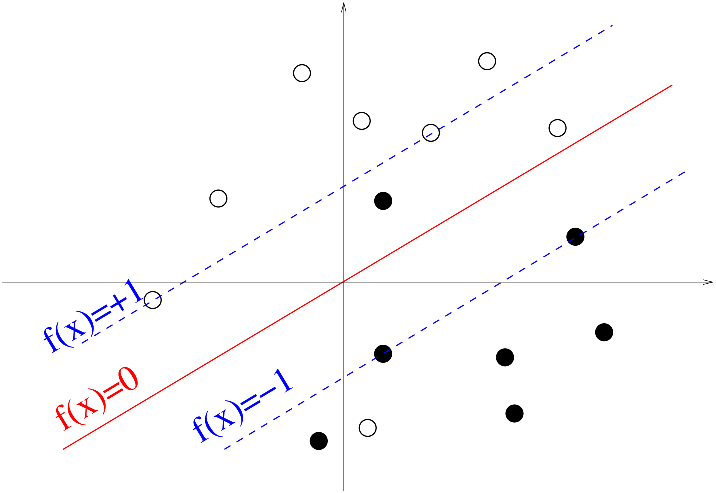


- Case 3: $\boldsymbol{\alpha}_i^\star = 0$;
- $[\nabla q(\boldsymbol{\alpha}^\star)]_i \geq 0$;
- $\Rightarrow y_i f(\mathbf{x}_i) \geq 1$.

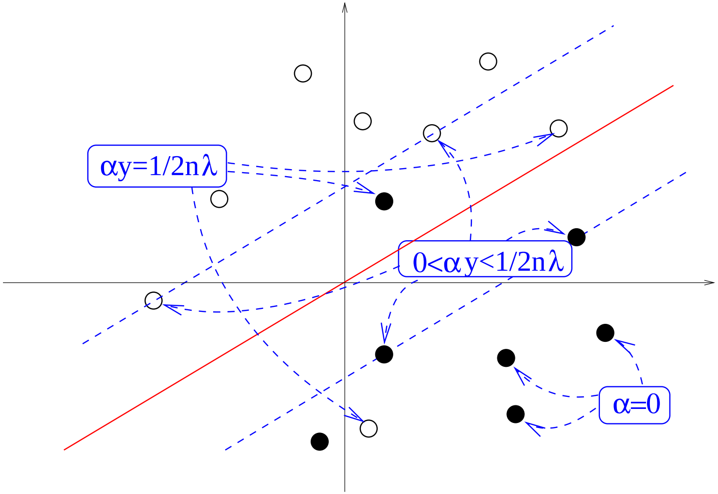
Geometric interpretation



Geometric interpretation



Geometric interpretation



Support vectors

Consequence of KKT conditions

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

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

where SV is the set of support vectors.

Consequences

- The solution is sparse in $\boldsymbol{\alpha}$, 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 λ as follows:

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

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

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

  subject to:

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

- 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:

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

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

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

  subject to:

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

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

Part 4

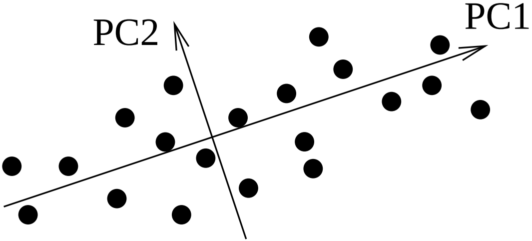
Kernel Methods
Unsupervised Learning

Outline

Principal Component Analysis (PCA)

Classical setting

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



Principal Component Analysis (PCA)

Formalization

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

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

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

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

Principal Component Analysis (PCA)

Formalization

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

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

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

$$\mathbf{w}_i = \underset{\mathbf{w}\perp\{\mathbf{w}_1,\ldots,\mathbf{w}_{i-1}\}}{\arg\max}\ \hat{var}\left(h_{\mathbf{w}}\right) \quad \text{s.t.} \quad \|\mathbf{w}\| = 1.$$

Principal Component Analysis (PCA)

Solution

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

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

- The solutions of:

$$\mathbf{w}_i = \underset{\mathbf{w} \perp \{\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}\}}{\arg\max} \; \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} \quad \text{s.t.} \quad \|\mathbf{w}\| = 1$$

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:

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

- The solutions of:

$$\mathbf{w}_i = \underset{\mathbf{w} \perp \{\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}\}}{\arg\max} \; \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} \;\; \text{s.t.} \;\; \|\mathbf{w}\| = 1$$

are the successive eigenvectors of $\mathbf{X}^\top \mathbf{X}$, 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 $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ 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.:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \quad \Longrightarrow \quad \frac{1}{n}\sum_{i=1}^{n}\varphi(\mathbf{x}_i) = 0.$$

- The orthogonal projection onto a direction $f \in \mathcal{H}$ is the function $h_f : \mathcal{X} \to \mathbb{R}$ defined by:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \frac{\mathbf{w}}{\|\mathbf{w}\|} \quad \Longrightarrow \quad h_f(\mathbf{x}) = \left\langle \varphi(\mathbf{x}), \frac{f}{\|f\|_{\mathcal{H}}} \right\rangle_{\mathcal{H}}.$$

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:

$$\hat{var}\left(h_{\mathbf{w}}\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left(\mathbf{x}_i^\top \mathbf{w}\right)^2}{\|\,\mathbf{w}\,\|^2} \quad \Longrightarrow \quad \hat{var}\left(h_f\right) := \frac{1}{n}\sum_{i=1}^{n}\frac{\langle \varphi(\mathbf{x}_i), f\rangle_{\mathcal{H}}^2}{\|\,f\,\|_{\mathcal{H}}^2}.$$

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

$$f_i = \underset{f \perp \{f_1,\ldots,f_{i-1}\}}{\arg\max}\ \hat{var}\left(h_f\right) \quad \text{s.t.} \quad \|f\|_{\mathcal{H}} = 1.$$

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:

$$\hat{var}\left(h_{\mathbf{w}}\right) = \frac{1}{n} \sum_{i=1}^{n} \frac{\left(\mathbf{x}_i^\top \mathbf{w}\right)^2}{\| \mathbf{w} \|^2} \qquad \Longrightarrow \qquad \hat{var}\left(h_f\right) := \frac{1}{n} \sum_{i=1}^{n} \frac{f(\mathbf{x}_i)^2}{\| f \|_{\mathcal{H}}^2}.$$

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

$$f_i = \underset{f \perp \{f_1, \ldots, f_{i-1}\}}{\arg\max} \sum_{i=1}^{n} f(\mathbf{x}_i)^2 \quad \text{s.t.} \quad \|f\|_{\mathcal{H}} = 1.$$

Sanity check: kernel PCA with linear kernel = PCA

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

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

  endowed with the norm $\| f_{\mathbf{w}} \|_{\mathcal{H}} = \| \mathbf{w} \|_{\mathbb{R}^d}$.
- Therefore we can write:

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

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

Kernel Principal Component Analysis (PCA)

Solution

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

$$f_i = \underset{f \perp \{f_1,\ldots,f_{i-1}\}}{\arg\max} \sum_{i=1}^{n} f(\mathbf{x}_i)^2 \quad \text{s.t.} \quad \|f\|_{\mathcal{H}} = 1.$$

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

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

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

Kernel Principal Component Analysis (PCA)

- Therefore we have:

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

- Similarly:

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

- and

$$\langle f_i, f_j\rangle_{\mathcal{H}} = \boldsymbol{\alpha}_i^\top \mathbf{K}\boldsymbol{\alpha}_j.$$

Kernel Principal Component Analysis (PCA)

Solution

Kernel PCA maximizes in $\boldsymbol{\alpha}$ the function:

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

under the constraints:

$$\begin{cases} \boldsymbol{\alpha}_i^\top \mathbf{K} \boldsymbol{\alpha}_j & = & 0 \quad \text{for } j = 1, \ldots, i-1 \\ \boldsymbol{\alpha}_i^\top \mathbf{K} \boldsymbol{\alpha}_i & = & 1 \end{cases}.$$

Kernel Principal Component Analysis (PCA)

Solution

- Compute the eigenvalue decomposition of the kernel matrix $\mathbf{K} = \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top$, with eigenvalues $\Delta_1 \geq \ldots \geq \Delta_n \geq 0$.
- After a change of variable $\boldsymbol{\beta} = \mathbf{K}^{1/2}\boldsymbol{\alpha}$ (with $\mathbf{K}^{1/2} = \mathbf{U}\boldsymbol{\Delta}^{1/2}\mathbf{U}^\top$),

$$\boldsymbol{\beta}_i = \underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\arg\max}\ \boldsymbol{\beta}^\top \mathbf{K}\boldsymbol{\beta},$$

  under the constraints:

$$\begin{cases} \boldsymbol{\beta}_i^\top \boldsymbol{\beta}_j & = & 0 \quad \text{for } j = 1, \ldots, i-1\,. \\ \boldsymbol{\beta}_i^\top \boldsymbol{\beta}_i & = & 1 \end{cases}$$

- Thus, $\boldsymbol{\beta}_i = \mathbf{u}_i$ (i-th eigenvector) is a solution!
- Finally, $\boldsymbol{\alpha}_i = \frac{1}{\sqrt{\boldsymbol{\Delta}_i}}\mathbf{u}_i$.

Kernel Principal Component Analysis (PCA)

Summary

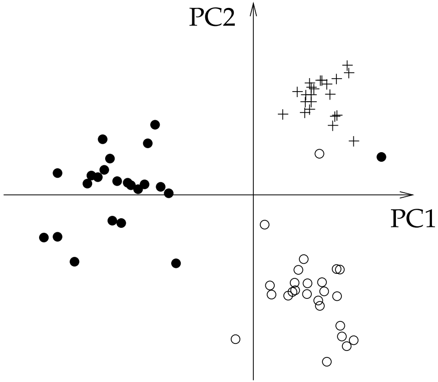
1. Center the Gram matrix
2. Compute the first eigenvectors $(\mathbf{u}_i, \Delta_i)$
3. Normalize the eigenvectors $\boldsymbol{\alpha}_i = \mathbf{u}_i / \sqrt{\Delta_i}$
4. The projections of the points onto the i-th eigenvector is given by $\mathbf{K}\boldsymbol{\alpha}_i$

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

Example



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

Outline

The K-means algorithm

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

Optimization point of view

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

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

K-means alternates between two steps:

1. **cluster assignment:**
   Given fixed $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$, assign each $\mathbf{x}_i$ to its closest centroid

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

The K-means algorithm

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

Optimization point of view

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

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

K-means alternates between two steps:

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

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

The K-means algorithm

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

Optimization point of view

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

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

K-means alternates between two steps:

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

$$\Leftrightarrow \forall j, \quad \boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{x}_i \quad \text{with} \quad C_j = \{i : s_i = j\}.$$

The kernel K-means algorithm

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

The kernel K-means algorithm

We may now modify the objective to operate in a RKHS. Given data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ 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

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

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

Proposition

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

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

The kernel K-means algorithm

Proof

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

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

The kernel K-means algorithm

Given now the objective,

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

we know that given assignments s_i, the optimal $\boldsymbol{\mu}_j$ 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 $s_1, \dots, s_n$, update the centroids

$$\forall j, \quad \boldsymbol{\mu}_j = \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{H}} \sum_{i : s_i = j} \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}\|_{\mathcal{H}}^2.$$

The kernel K-means algorithm

Given now the objective,

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

we know that given assignments s_i, the optimal $\boldsymbol{\mu}_j$ 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 $s_1, \dots, s_n$, update the centroids

$$\Leftrightarrow \forall j, \quad \boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \varphi(\mathbf{x}_i).$$

The kernel K-means algorithm

Given now the objective,

$$\min_{\substack{\boldsymbol{\mu}_j \in \mathcal{H} \quad \text{for} \quad j=1,\ldots,k \\ s_i \in \{1,\ldots,k\} \quad \text{for} \quad i=1,\ldots,n}} \sum_{i=1}^{n} \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_{s_i}\|_{\mathcal{H}}^2,$$

we know that given assignments s_i, the optimal $\boldsymbol{\mu}_j$ 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 $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$, assign each $\mathbf{x}_i$ to its closest centroid: $\forall\ i$,

$$s_i \in \operatorname*{argmin}_{s \in \{1,\ldots,k\}} \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_s\|_{\mathcal{H}}^2.$$

The kernel K-means algorithm

Given now the objective,

$$\min_{\substack{\boldsymbol{\mu}_j \in \mathcal{H} \quad \text{for} \quad j=1,\ldots,k \\ s_i \in \{1,\ldots,k\} \quad \text{for} \quad i=1,\ldots,n}} \sum_{i=1}^{n} \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_{s_i}\|_{\mathcal{H}}^2,$$

we know that given assignments s_i, the optimal $\boldsymbol{\mu}_j$ 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 $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$, assign each $\mathbf{x}_i$ to its closest centroid: $\forall\ i$,

$$s_i \in \operatorname*{argmin}_{s \in \{1,\ldots,k\}} \left\| \varphi(\mathbf{x}_i) - \frac{1}{|C_s|} \sum_{j \in C_s} \varphi(\mathbf{x}_j) \right\|_{\mathcal{H}}^2 \qquad (C_s \text{ is from step 1}).$$

The kernel K-means algorithm

Given now the objective,

$$\min_{\substack{\boldsymbol{\mu}_j \in \mathcal{H} \quad \text{for} \quad j=1,\ldots,k \\ s_i \in \{1,\ldots,k\} \quad \text{for} \quad i=1,\ldots,n}} \sum_{i=1}^{n} \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_{s_i}\|_{\mathcal{H}}^2,$$

we know that given assignments s_i, the optimal $\boldsymbol{\mu}_j$ 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 $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$, assign each $\mathbf{x}_i$ to its closest centroid: $\forall\ i$,

$$s_i \in \operatorname*{argmin}_{s \in \{1,\ldots,k\}} \left(K(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{|C_s|} \sum_{j \in C_s} K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{|C_s|^2} \sum_{j,l \in C_s} K(\mathbf{x}_j, \mathbf{x}_l) \right).$$

The kernel K-means algorithm, equivalent objective

Note that all operations are performed by manipulating kernel values $K(\mathbf{x}_i, \mathbf{x}_j)$ only. Implicitly, we are optimizing in fact

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

or, equivalently,

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

Then, notice that

$$\sum_{i=1}^{n} \frac{1}{|C_{s_i}|^2} \sum_{j,l \in C_{s_i}} K(\mathbf{x}_j, \mathbf{x}_l) = \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{i,j \in C_l} K(\mathbf{x}_i, \mathbf{x}_j).$$

The kernel K-means algorithm, equivalent objective

Note that all operations are performed by manipulating kernel values $K(\mathbf{x}_i, \mathbf{x}_j)$ only. Implicitly, we are optimizing in fact

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

or, equivalently,

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

and

$$\sum_{i=1}^{n} \frac{1}{|C_{s_i}|} \sum_{j \in C_{s_i}} K(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{i,j \in C_l} K(\mathbf{x}_i, \mathbf{x}_j).$$

The kernel K-means algorithm, equivalent objective

Then, after removing the constant terms $K(\mathbf{x}_i, \mathbf{x}_i)$, we obtain:

Proposition

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

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

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 $K(\mathbf{x}_i, \mathbf{x}_i)$, we obtain:

Proposition

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

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

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

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

and we introduce

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

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

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

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

Spectral clustering algorithms

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

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

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

Spectral clustering algorithms

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

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

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

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!

Question

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

Spectral clustering algorithms

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

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

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

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 $\mathbf{A}$, every row of $\mathbf{A}$ has a single non-zero entry ⇒ compute the maximum entry of every row of $\mathbf{Z}^\star$.

Spectral clustering algorithms

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

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

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

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 2

Normalize the rows of $\mathbf{Z}^\star$ to have unit ℓ_2-norm, and apply the traditional K-means algorithm on the rows. This is called spectral clustering.

Spectral clustering algorithms

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

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

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

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 3

Choose another variant of the previous procedures.

Outline

1. Kernels and RKHS

2. Kernel tricks

3. Kernel Methods: Supervised Learning

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

Formulation

Assuming that the datasets are centered, we want to maximize

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

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

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

Canonical Correlation Analysis (CCA)

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

Formulation

Assuming that the datasets are centered, we want to maximize

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

It is possible to show that this is an generalized eigenvalue problem (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,

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

can be formulated, after removing the scaling ambiguity, as

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

Then, there exists λ_a and λ_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

$$-\mathbf{X}^\top \mathbf{Y}\mathbf{w}_b + \lambda_a \mathbf{X}^\top \mathbf{X}\mathbf{w}_a = 0$$
$$-\mathbf{Y}^\top \mathbf{X}\mathbf{w}_a + \lambda_b \mathbf{Y}^\top \mathbf{Y}\mathbf{w}_b = 0$$

Multiply first equality by $\mathbf{w}_a^\top$ and second equality by $\mathbf{w}_b^\top$; subtract the two resulting equalities and we get

$$\lambda_a \mathbf{w}_a^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}_a = \lambda_b \mathbf{w}_b^\top \mathbf{Y}^\top \mathbf{Y}\mathbf{w}_b = \lambda_a = \lambda_b = \lambda,$$

and then, we obtain the generalized eigenvalue problem:

$$\begin{bmatrix} 0 & \mathbf{X}^\top \mathbf{Y} \\ \mathbf{Y}^\top \mathbf{X} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{X}^\top \mathbf{X} & 0 \\ 0 & \mathbf{Y}^\top \mathbf{Y} \end{bmatrix} \begin{bmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{bmatrix}$$

Canonical Correlation Analysis (CCA)

Let us define

$$\boldsymbol{\Sigma}_A = \begin{bmatrix} 0 & \mathbf{X}^\top \mathbf{Y} \\ \mathbf{Y}^\top \mathbf{X} & 0 \end{bmatrix}, \quad \boldsymbol{\Sigma}_B = \begin{bmatrix} \mathbf{X}^\top \mathbf{X} & 0 \\ 0 & \mathbf{Y}^\top \mathbf{Y} \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{bmatrix}$$

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

$$\boldsymbol{\Sigma}_B^{-1/2} \boldsymbol{\Sigma}_A \mathbf{w} = \lambda \boldsymbol{\Sigma}_B^{1/2} \mathbf{w}$$

which is also equivalent to the eigenvalue problem

$$\boldsymbol{\Sigma}_B^{-1/2} \boldsymbol{\Sigma}_A \boldsymbol{\Sigma}_B^{-1/2} (\boldsymbol{\Sigma}_B^{1/2} \mathbf{w}) = \lambda (\boldsymbol{\Sigma}_B^{1/2} \mathbf{w}).$$

Kernel Canonical Correlation Analysis

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

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

where $\varphi_a : \mathcal{X} \to \mathcal{H}_a$ and $\varphi_b : \mathcal{X} \to \mathcal{H}_b$ 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

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

Kernel Canonical Correlation Analysis

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

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

where $\varphi_a : \mathcal{X} \to \mathcal{H}_a$ and $\varphi_b : \mathcal{X} \to \mathcal{H}_b$ 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

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

Kernel Canonical Correlation Analysis

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

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

which is equivalent to

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

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

Equivalent formulation

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

Kernel Canonical Correlation Analysis

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

- 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

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

- 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

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

- 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 $\mathbf{K}_a$ and $\mathbf{K}_b$ are invertible, make the change of variable $\boldsymbol{\alpha}' = \mathbf{K}_a \boldsymbol{\alpha}$ and $\boldsymbol{\beta}' = \mathbf{K}_b \boldsymbol{\beta}$, and we obtain the equivalent formulation

$$\max_{\boldsymbol{\alpha}'\in\mathbb{R}^n,\boldsymbol{\beta}'\in\mathbb{R}^n} \boldsymbol{\alpha}'^\top \boldsymbol{\beta}' \;\text{ s.t. }\; \boldsymbol{\alpha}'^\top \boldsymbol{\alpha}' = 1 \;\text{ and }\; \boldsymbol{\beta}'^\top \boldsymbol{\beta}' = 1.$$

The function is maximized for any $\boldsymbol{\alpha}' = \boldsymbol{\beta}'$ in $\mathbb{R}^n$.

Kernel Canonical Correlation Analysis

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

- 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 $\mathbf{K}_a$ and $\mathbf{K}_b$ are invertible, make the change of variable $\boldsymbol{\alpha}' = \mathbf{K}_a\boldsymbol{\alpha}$ and $\boldsymbol{\beta}' = \mathbf{K}_b\boldsymbol{\beta}$, and we obtain the equivalent formulation

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

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

Spurious correlations

Spurious correlations are bad:

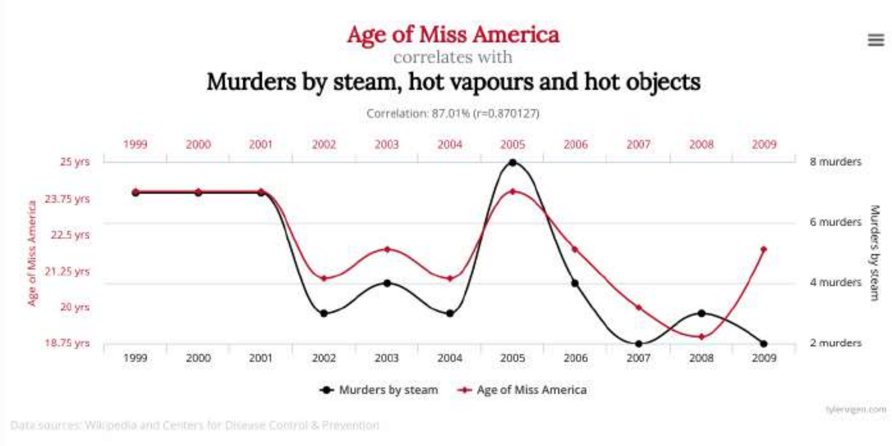
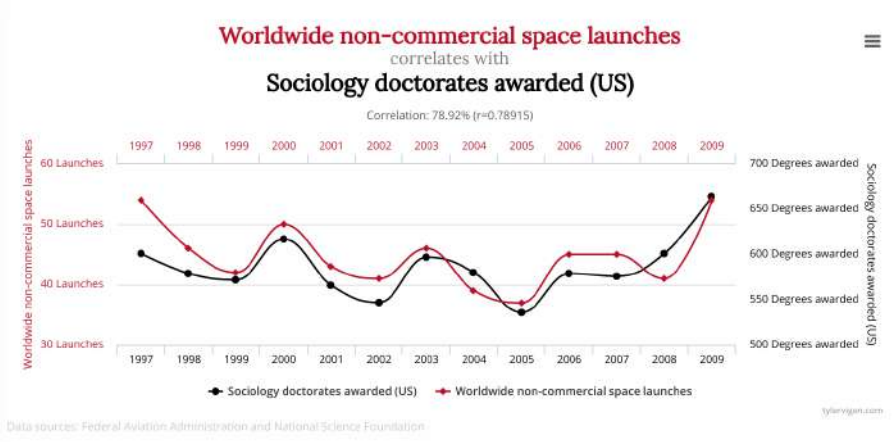


Figure: `http://www.tylervigen.com/`.

Spurious correlations

Spurious correlations are bad:



Data sources: Federal Aviation Administration and National Science Foundation

Figure: `http://www.tylervigen.com/`.

Kernel Canonical Correlation Analysis

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

- 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

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

- 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 $\boldsymbol{\alpha}^\top\mathbf{K}_a^2\boldsymbol{\alpha} = 1$ by

$$(1-\tau)\boldsymbol{\alpha}^\top\mathbf{K}_a^2\boldsymbol{\alpha} + \tau\underbrace{\boldsymbol{\alpha}^\top\mathbf{K}_a\boldsymbol{\alpha}}_{\|f_a\|^2_{\mathcal{H}_a}} = 1,$$

and do the same for $\boldsymbol{\beta}^\top\mathbf{K}_b^2\boldsymbol{\beta} = 1$.

Application of kernel CCA

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

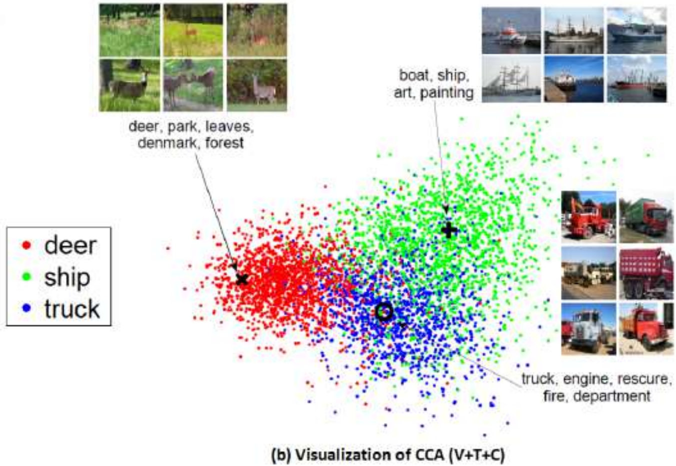


Figure: Figure from Gong and Lazebnik, 2014.

Part 5

The Kernel Jungle

Outline

1. Kernels and RKHS

2. Kernel tricks

3. Kernel Methods: Supervised Learning

4. Kernel Methods: Unsupervised 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.

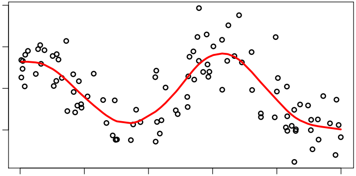
Outline

5. 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

$$\min_f \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda \int \left(f''(t)\right)^2 dt$$



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

A simple example

Let

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

endowed with the bilinear form:

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

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

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

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

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda \int_0^1 \left(f'(t)\right)^2 dt$$

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

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

Technical remark

Definition

Let $I \subset \mathbb{R}$ an interval. A function $f : I \to \mathbb{R}$ 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:

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

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:

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

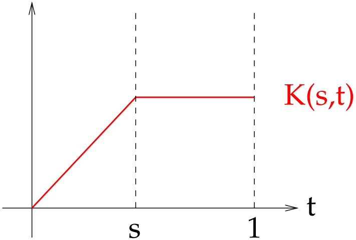
  showing that f is absolutely continuous and $f' = g$ almost everywhere; in particular, $f' \in L^2[0,1]$.
- Finally, $f(0) = \lim_n f_n(0) = 0$, therefore $f \in \mathcal{H}$ and

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

Proof (4/5)

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

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



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

Proof (5/5)

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. $\square$

Generalization

Theorem

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

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

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

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:

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

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

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

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:

$$\begin{aligned} f(x) &= \langle D^* D k_x, f \rangle_{L^2([0,1])} \\ &= \langle D k_x, Df \rangle_{L^2([0,1])} \\ &= \int_0^1 k_x'(u) f'(u) du \end{aligned}$$

- The solution is

$$k_x'(u) = \mathbf{1}_{[0,x]}(u)$$

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

5 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. $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ 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:

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

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 ν 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:

$$\forall f \in L^2_\nu(\mathcal{X}), \quad (L_K f)(\mathbf{x}) = \int K(\mathbf{x}, \mathbf{t}) f(\mathbf{t})\, d\nu(\mathbf{t}).$$

Lemma

If K is a Mercer kernel, then L_K is a compact and bounded linear operator over $L^2_\nu(\mathcal{X})$, self-adjoint and positive.

Proof (1/6)

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 $(\mathbf{x}_1, \mathbf{x}_1) \in \mathcal{X}^2$:

$$\begin{aligned}
|(L_K f)(\mathbf{x}_1) - (L_K f)(\mathbf{x}_2)| &= \left| \int (K(\mathbf{x}_1, \mathbf{t}) - K(\mathbf{x}_2, \mathbf{t})) f(\mathbf{t}) \, d\nu(\mathbf{t}) \right| \\
&= \langle K_{\mathbf{x}_1} - K_{\mathbf{x}_2}, f \rangle_{L^2_\nu(\mathcal{X})} \\
&\leq \| K_{\mathbf{x}_1} - K_{\mathbf{x}_2} \|_{L^2_\nu(\mathcal{X})} \| f \|_{L^2_\nu(\mathcal{X})} \\
&\qquad\qquad \text{(Cauchy-Schwarz)} \\
&\leq \sqrt{\nu(\mathcal{X})} \max_{\mathbf{t} \in \mathcal{X}} | K(\mathbf{x}_1, \mathbf{t}) - K(\mathbf{x}_2, \mathbf{t}) | \, \| f \|_{L^2_\nu(\mathcal{X})} .
\end{aligned}$$

K being continuous and $\mathcal{X}$ compact, K is uniformly continuous, therefore $L_K f$ is continuous. In particular, $L_K f \in L^2_\nu(\mathcal{X})$ (with the slight abuse of notation $\mathcal{C}(\mathcal{X}) \subset L^2_\nu(\mathcal{X})$). $\square$

Proof (2/6)

L_K is linear and continuous

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

$$\begin{aligned} |(L_K f)(\mathbf{x})| &= \left| \int K(\mathbf{x}, \mathbf{t}) f(\mathbf{t}) \, d\nu(\mathbf{t}) \right| \\ &\leq \sqrt{\nu(\mathcal{X})} \max_{\mathbf{t} \in \mathcal{X}} |K(\mathbf{x}, \mathbf{t})| \, \| f \|_{L^2_\nu(\mathcal{X})} \\ &\leq \sqrt{\nu(\mathcal{X})} C_K \| f \|_{L^2_\nu(\mathcal{X})}. \end{aligned}$$

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

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

Proof (3/6)

Criterion for compactness

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

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

Ascoli Theorem

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

Proof (4/6)

L_K is compact

Let $(f_n)_{n\geq 0}$ be a bounded sequence of $L^2_\nu(\mathcal{X})$ ($\| f_n \|_{L^2_\nu(\mathcal{X})} < M$ for all n). The sequence $(L_K f_n)_{n\geq 0}$ is a sequence of continuous functions, uniformly bounded because:

$$\| L_K f_n \|_\infty \leq \sqrt{\nu(\mathcal{X})} C_K \| f_n \|_{L^2_\nu(\mathcal{X})} \leq \sqrt{\nu(\mathcal{X})} C_K M \,.$$

It is equicontinuous because:

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

By Ascoli theorem, we can extract a sequence uniformly convergent in $C(\mathcal{X})$, and therefore in $L^2_\nu(\mathcal{X})$. $\square$

Proof (5/6)

L_K is self-adjoint

K being symmetric, we have for all $f, g \in L^2_\nu(\mathcal{X})$:

$$\begin{aligned}\langle f, Lg\rangle_{L^2_\nu(\mathcal{X})} &= \int f(\mathbf{x})\,(Lg)(\mathbf{x})\,d\nu(\mathbf{x}) \\ &= \int\int f(\mathbf{x})\,g(\mathbf{t})\,K(\mathbf{x},\mathbf{t})\,d\nu(\mathbf{x})\,d\nu(\mathbf{t}) \quad \text{(Fubini)} \\ &= \langle Lf, g\rangle_{L^2_\nu(\mathcal{X})}.\end{aligned}$$

Proof (6/6)

L_K is positive

We can approximate the integral by finite sums:

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

because K is positive definite. □

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 ψ_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, ν 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 ψ_k are continuous, and for any $\mathbf{x}, \mathbf{t} \in \mathcal{X}$:

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

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

---

[a] i.e., $\nu(U) > 0$ for any nonempty open set $U \subset \mathcal{X}$

Proof of Mercer's Theorem (1/6)

For any $k \geq 1$ such that $\lambda_k > 0$, $\psi_k \in \mathcal{H}$ (RKHS of K)

If $\lambda_k > 0$, we have

$$\begin{aligned} \forall \mathbf{x} \in \mathcal{X}\,, \quad \psi_k(\mathbf{x}) &= \frac{1}{\lambda_k} L_K \psi_k(\mathbf{x}) \\ &= \frac{1}{\lambda_k} \int K(\mathbf{x}, \mathbf{t}) \psi_k(\mathbf{t}) d\nu(\mathbf{t}) \\ &= \lim_{n \to +\infty} \underbrace{\frac{\nu(\mathcal{X})}{\lambda_k n} \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{t}_i) \psi_k(\mathbf{t}_i)}_{h_n(\mathbf{x})} \end{aligned}$$

for a set $\mathbf{t}_1, \mathbf{t}_2, \ldots$ conveniently chosen. Besides, $h_n \in \mathcal{H}$ for any $n \in \mathbb{N}$ and, for any $n, m \in \mathbb{N}$,

$$\langle h_n, h_m \rangle_{\mathcal{H}} = \frac{\nu(\mathcal{X})^2}{\lambda_k^2 nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \psi_k(\mathbf{t}_i) \psi_k(\mathbf{t}_j) K(\mathbf{t}_i, \mathbf{t}_j)\,.$$

Proof of Mercer's Theorem (2/6)

For any $k \geq 1$ such that $\lambda_k > 0$, $\psi_k \in \mathcal{H}$ (cont.)

Therefore,

$$\lim_{n,m \to +\infty} \langle h_n, h_m \rangle_{\mathcal{H}} = \frac{1}{\lambda_k^2} \int \int K(\mathbf{t}, \mathbf{t}') \psi_k(\mathbf{t}) \psi_k(\mathbf{t}') d\nu(\mathbf{t}) d\nu(\mathbf{t}') := R \,,$$

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 $\mathbf{x} \in \mathcal{X}$,

$$h(x) = \lim_{n \to +\infty} h_n(\mathbf{x}) = \psi_k(\mathbf{x}) \,,$$

and finally $\psi_k = h \implies \psi_k \in \mathcal{H}$. $\square$

Proof of Mercer's Theorem (3/6)

$\left\{\sqrt{\lambda_k}\psi_k \,:\, \lambda_k > 0\right\}$ in an orthonormal system (ONS) of $\mathcal{H}$

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

$$\begin{aligned}
\left\langle \sqrt{\lambda_i}\psi_i, \sqrt{\lambda_j}\psi_j \right\rangle_{\mathcal{H}} &= \left\langle \frac{1}{\sqrt{\lambda_i}} \int K_{\mathbf{t}}\psi_i(\mathbf{t})d\nu(\mathbf{t}), \psi_i, \sqrt{\lambda_j}\psi_j \right\rangle_{\mathcal{H}} \\
&= \sqrt{\frac{\lambda_j}{\lambda_i}} \int \langle K_{\mathbf{t}}, \psi_j \rangle_H \psi_i(\mathbf{t})d\nu(\mathbf{t}) \\
&= \sqrt{\frac{\lambda_j}{\lambda_i}} \int \psi_j(\mathbf{t})\psi_i(\mathbf{t})d\nu(\mathbf{t}) \\
&= \sqrt{\frac{\lambda_j}{\lambda_i}} \langle \psi_i, \psi_j \rangle_{L^2_\nu(\mathcal{X})} \\
&= \delta_{i,j} \,. \qquad \square
\end{aligned}$$

Proof of Mercer's Theorem (4/6)

For any $\mathbf{x} \in \mathcal{X}, \sum_{k:\lambda_k>0} \lambda_k \psi_k(\mathbf{x})^2 \leq C_K$

For any $\mathbf{x} \in \mathcal{X}$, $K_x \in \mathcal{H}$ and $\|K_x\|_{\mathcal{H}}^2 = K(\mathbf{x},\mathbf{x}) \leq C_K$.
Therefore, since $\left\{\sqrt{\lambda_k}\psi_k \,:\, \lambda_k > 0\right\}$ is an ONS of $\mathcal{H}$:

$$\begin{aligned} C_K &\geq \|K_x\|_{\mathcal{H}}^2 \\ &\geq \sum_{k:\lambda_k>0} \left\langle K_{\mathbf{x}}, \sqrt{\lambda_k}\psi_k \right\rangle_{\mathcal{H}}^2 \\ &= \sum_{k:\lambda_k>0} \lambda_k \psi_k(\mathbf{x})^2 \,. \qquad \square \end{aligned}$$

Proof of Mercer's Theorem (5/6)

For any $\mathbf{x} \in \mathcal{X}, \mathbf{t} \rightarrow \sum_i \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{t})$ convergences uniformly to a continuous function $g_{\mathbf{x}}$

For any fixed $\mathbf{x} \in \mathcal{X}$, we therefore have, for any $\mathbf{t} \in \mathcal{X}$ (restricting the sum to the indices $i \geq 1$ such that $\lambda_i > 0$):

$$\begin{aligned} \sum_{i=m}^{m+\ell} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{t}) &\leq \left(\sum_{i=m}^{m+\ell} \lambda_i \psi_i(\mathbf{x})^2 \right)^{\frac{1}{2}} \left(\sum_{i=m}^{m+\ell} \lambda_i \psi_i(\mathbf{t})^2 \right)^{\frac{1}{2}} \\ &\leq C_K \left(\sum_{i=m}^{m+\ell} \lambda_i \psi_i(\mathbf{x})^2 \right)^{\frac{1}{2}} , \end{aligned}$$

which tends to 0 uniformly in $\mathbf{t} \in \mathcal{X}$. Therefore the series of functions $\mathbf{t} \rightarrow \sum_i \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{t})$ is uniformly Cauchy, continuous, and therefore convergences uniformly to a continuous function $g_{\mathbf{x}}$.

Proof of Mercer's Theorem (6/6)

$K_{\mathbf{x}} = g_{\mathbf{x}}$ in $L_2(\nu)$

On the other hand, we can expand $K_{\mathbf{x}}$ over the ONB $\{\psi_k, k \geq 1\}$ of $L^2_\nu(\mathcal{X})$:

$$\begin{aligned} K_{\mathbf{x}} &= \sum_{k \geq 1} \langle K_{\mathbf{x}}, \psi_k \rangle_{L^2_\nu(\mathcal{X})} \psi_k \\ &= \sum_{k \geq 1} (L\psi_k)(\mathbf{x})\psi_k \\ &= \sum_{k \geq 1} \lambda_k \psi_k(\mathbf{x})\psi_k \\ &= \sum_{k \geq 1 \,:\, \lambda_k > 0} \lambda_k \psi_k(\mathbf{x})\psi_k \,, \end{aligned}$$

therefore $K_{\mathbf{x}} = g_{\mathbf{x}}$ in $L_2(\nu)$, i.e., $\|K_{\mathbf{x}} - g_{\mathbf{x}}\|_{L_2(\nu)} = 0$.

Proof of Mercer's Theorem (5/5)

Conclusion

Since ν in nondegenerate, and both $K_{\mathbf{x}}$ and $g_{\mathbf{x}}$ are continuous, this implies

$$\forall \mathbf{t} \in \mathcal{X}\,, \quad K_{\mathbf{x}}(\mathbf{t}) = g_{\mathbf{x}}(\mathbf{t}) = \sum_i \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{t})\,,$$

and the convergence is uniform in $\mathcal{X} \times \mathcal{X}$ because K is continuous.

Mercer kernels as inner products

Let ℓ^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

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

is well defined, continuous, and satisfies

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

Proof of the corollary

- By Mercer theorem we see that for all $\mathbf{x} \in \mathcal{X}$, $\sum \lambda_k \psi_k^2(\mathbf{x})$ converges to $K(\mathbf{x}, \mathbf{x}) < \infty$, therefore $\Phi(\mathbf{x}) \in \ell^2$.
- The continuity of Φ results from:

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

Summary

- This proof extends the proof valid when $\mathcal{X}$ is finite.
- This is a constructive proof, developed by Mercer (1905).
- The eigensystem (λ_k and ψ_k) depend on the choice of the measure $d\nu(\mathbf{x})$: different ν'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:

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

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 $d\nu(\mathbf{x}) = d\mathbf{x}$
- Let a p.d. kernel on $\mathcal{X}$ of the form

$$K(\mathbf{x},\mathbf{t}) = \kappa(\mathbf{x}-\mathbf{t}) \; ,$$

  where $\kappa : \mathbb{R} \rightarrow \mathbb{R}$ is continuous and 1-periodic.
- To write Mercer's expansion we need to find the eigenfunctions of L_K by solving

$$(L_K\psi)(\mathbf{x}) = \int_0^1 \kappa(\mathbf{x}-\mathbf{t})\,\psi(\mathbf{t})\,d\mathbf{t} = \lambda\psi(\mathbf{x})$$

Example 1: $[0, 1]$ (2/6)

Lemma

Let $(\psi_n)_{n\in\mathbb{N}}$ *be the Fourier ONB of* $L^2([0,1])$ *given by* $\psi_0(\mathbf{x}) = 1$ *and*

$$\forall n \geq 1\,, \quad \begin{cases} \psi_{2n-1}(\mathbf{x}) & = \sqrt{2}\sin(2\pi n\mathbf{x})\,, \\ \psi_{2n}(\mathbf{x}) & = \sqrt{2}\cos(2\pi n\mathbf{x})\,. \end{cases}$$

Let the Fourier expansion of κ *be*[a]

$$\forall \mathbf{x} \in [0,1]\,, \quad \kappa(\mathbf{x}) = \sum_{n=0}^{\infty} \hat{\kappa}_{2n}\psi_{2n}(\mathbf{x})\,.$$

Then for any $n \in \mathbb{N}$, ψ_n *is an eigenfunction of* L_K *with eigenvalues* $\hat{\kappa}_0$ *for* ψ_0 *and* $\hat{\kappa}_{2n}/\sqrt{2}$ *for* ψ_{2n-1} *and* ψ_{2n}.

---

[a] K symmetric $\implies$ κ 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 $\int_0^1 \psi_i(\mathbf{x})\psi_j(\mathbf{x})d\mathbf{x} = \delta_{ij}$.
- By trigonometric expansion of $\sin(a+b)$ and $\cos(a+b)$, show that

$$\begin{cases} \psi_{2n}(\mathbf{x}-\mathbf{t}) & = \frac{1}{\sqrt{2}}\left[\psi_{2n}(\mathbf{x})\psi_{2n}(\mathbf{t}) + \psi_{2n-1}(\mathbf{x})\psi_{2n-1}(\mathbf{t})\right], \\ \psi_{2n-1}(\mathbf{x}-\mathbf{t}) & = \frac{1}{\sqrt{2}}\left[\psi_{2n-1}(\mathbf{x})\psi_{2n}(\mathbf{t}) - \psi_{2n}(\mathbf{x})\psi_{2n-1}(\mathbf{t})\right]. \end{cases}$$

- Then direct computation of $L_K\psi_i$, e.g.,

$$\begin{aligned} L_K\psi_{2n}(\mathbf{x}) &= \sum_{\ell=0}^{\infty} \hat{\kappa}_{2\ell} \int_0^1 \psi_{2\ell}(\mathbf{x}-\mathbf{t})\psi_{2n}(\mathbf{t})d\mathbf{t} \\ &= \sum_{\ell=0}^{\infty} \frac{\hat{\kappa}_{2\ell}}{\sqrt{2}} \int_0^1 \left[\psi_{2\ell}(\mathbf{x})\psi_{2\ell}(\mathbf{t}) + \psi_{2\ell-1}(\mathbf{x})\psi_{2\ell-1}(\mathbf{t})\right]\psi_{2n}(\mathbf{t})d\mathbf{t} \\ &= \sum_{\ell=0}^{\infty} \frac{\hat{\kappa}_{2\ell}}{\sqrt{2}}\psi_{2\ell}(\mathbf{x})\delta_{n\ell} = \frac{\hat{\kappa}_{2n}}{\sqrt{2}}\psi_{2n}(\mathbf{x}). \quad \square \end{aligned}$$

Example 1: $[0, 1]$ (4/6)

Remark: Mercer's theorem is obviously correct. All ψ_k's are continuous, and for any $\mathbf{x}, \mathbf{t} \in [0, 1]$ the Mercer expansion of the kernel is:

$$\begin{aligned} K(\mathbf{x}, \mathbf{t}) &= \hat{\kappa}_0 + \sum_{n=1}^{\infty} \frac{\hat{\kappa}_{2n}}{\sqrt{2}} \left[\psi_{2n-1}(\mathbf{x})\psi_{2n-1}(\mathbf{t}) + \psi_{2n}(\mathbf{x})\psi_{2n}(\mathbf{t})\right] \\ &= \sum_{n=0}^{\infty} \hat{\kappa}_{2n}\psi_{2n}(\mathbf{x} - \mathbf{t}) \\ &= \kappa(\mathbf{x} - \mathbf{t}), \end{aligned} \tag{3}$$

with absolute and uniform convergence (because κ is continuous).

Example 1: $[0,1]$ (5/6)

Example: polynomial decay of eigenvalues

For any $\beta \in \mathbb{N}^*$, let

$$\begin{cases} \hat{\kappa}_0 & = 0\,, \\ \hat{\kappa}_{2n} & = \sqrt{2} n^{-2\beta} \text{ for } n \geq 1\,. \end{cases}$$

Then the corresponding kernel is

$$\forall \mathbf{x}, \mathbf{t} \in [0,1]\,, \quad K(\mathbf{x}, \mathbf{t}) = \frac{1}{(2\beta)!} B_{2\beta}(\mathbf{x} - \mathbf{t} - \lfloor \mathbf{x} - \mathbf{t} \rfloor)\,,$$

where $B_{2\beta}$ is the (2β)-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 exercice (check Fourier expansion of Bernoulli polynomials).

Example 1: $[0, 1]$ (6/6)

Example: exponential decay of eigenvalues

For any $\rho \in \mathbb{R}_+$, let

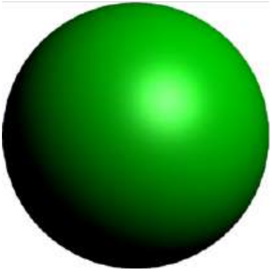
$$\begin{cases} \hat{\kappa}_0 & = 0\,, \\ \hat{\kappa}_{2n} & = e^{-\rho n} \text{ for } n \geq 1\,. \end{cases}$$

Then the corresponding kernel is

$$\forall \mathbf{x}, \mathbf{t} \in [0, 1]\,, \quad K(\mathbf{x}, \mathbf{t}) = \frac{\sqrt{2} e^{\rho} \cos\left(2\pi(\mathbf{x} - \mathbf{t})\right) - 1}{e^{2\rho} - 2e^{\rho} \cos\left(2\pi(\mathbf{x} - \mathbf{t})\right) + 1}\,.$$

Proof left as exercice (or check Bach, 2013, p.21).

Example 2: S^{d-1} (1/6)



- Consider the unit sphere in $\mathbb{R}^d$:

$$\mathcal{X} = S^{d-1} = \left\{ \mathbf{x} \in \mathbb{R}^d \ : \ \| \mathbf{x} \| = 1 \right\}$$

- Let ν be the Lebesgue measure on S^{d-1}. Note that:

$$\nu(S^{d-1}) = \frac{2\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)}$$

Example 2: S^{d-1} (2/6)

- Let a p.d. kernel on S^{d-1} of the form:

$$K\left(\mathbf{x},\mathbf{t}\right)=\varphi\left(\mathbf{x}^\top\mathbf{t}\right)\ ,$$

  where $\varphi:[-1,1]\to\mathbb{R}$ is continuous.

- To write Mercer's expansion we need to find the eigenfunctions by solving

$$\int_{S^{d-1}}\varphi\left(\mathbf{x}^\top\mathbf{t}\right)\psi\left(\mathbf{t}\right)d\nu(\mathbf{t})=\lambda\psi\left(\mathbf{x}\right)$$

- 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 Δ is the Laplacian operator on $\mathbb{R}^d$.

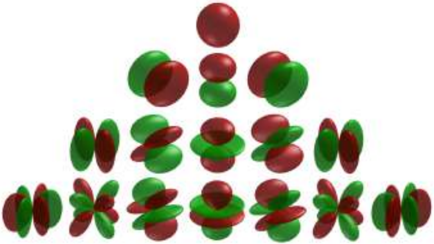
Example 2: S^{d-1} (3/6)

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

$$N(n,k) = dim\left(\mathcal{Y}_k(d)\right) = \frac{(2k+d-2)(k+d-3)!}{k!(d-2)!}.$$



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 $\mathbf{x} \in S^{d-1}$, $Y_k \in \mathcal{Y}_k(d)$ and $\varphi \in C\left([-1,1]\right)$,

$$\int_{S^{d-1}} \varphi\left(\mathbf{x}^\top \mathbf{t}\right) Y_k\left(\mathbf{t}\right) d\nu(\mathbf{t}) = \lambda_k Y_k\left(\mathbf{x}\right)$$

where

$$\lambda_k = \nu\left(S^{d-2}\right) \int_{-1}^{1} \varphi(t) P_k(d;t)(1-t^2)^{\frac{d-3}{2}} dt$$

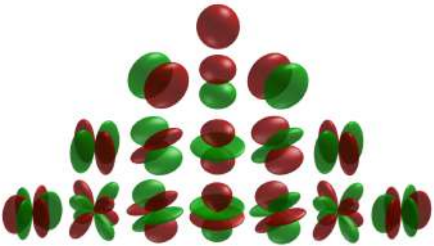
and $P_k(d;t)$ is the Legendre polynomial of degree k in dimension d.

When $\varphi \in C^k\left([-1,1]\right)$ we have Rodrigues rule (Müller, 1998, p.23):

$$\lambda_k = \nu\left(S^{d-2}\right) \frac{\Gamma\left(\frac{d-1}{2}\right)}{2^k \Gamma\left(k + \frac{d-1}{2}\right)} \int_{-1}^{1} \varphi^{(k)}(t) \left(1-t^2\right)^{k+\frac{d-3}{2}} dt$$

Example 2: S^{d-1} (5/6)

- For any $k \geq 0$, let $\{Y_{k,j}(d;\mathbf{x})\}_{j=1}^{N(d;k)}$ an orthonormal basis of $\mathcal{Y}_k(d)$
- Spherical harmonics $\left\{\{Y_{k,j}(d;\mathbf{x})\}_{j=1}^{N(d;k)}\right\}_{k=0}^{\infty}$ form an orthonormal basis for $L^2\left(S^{d-1}\right)$
- Therefore, for any kernel $K\left(\mathbf{x},\mathbf{t}\right) = \varphi\left(\mathbf{x}^\top\mathbf{t}\right)$ on S^{d-1} the Mercer eigenvalues are exactly the λ_k's, with corresponding orthonormal eigenfunctions $\{Y_{k,j}(d;\mathbf{x})\}_{j=1}^{N(d;k)}$.
- Note that eigenfunctions are the same for different φ's, only the eigenvalues change



Example 2: S^{d-1} (6/6)

- Take $d = 2$ and $K(\mathbf{x}, \mathbf{t}) = \left(1 + \mathbf{x}^\top \mathbf{t}\right)^2$ for $\mathbf{x}, \mathbf{t} \in S^1$
- 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:

$$\left(\frac{1}{\sqrt{2\pi}}, \frac{x_1}{\sqrt{\pi}}, \frac{x_2}{\sqrt{\pi}}, \frac{x_1 x_2}{\sqrt{\pi}}, \frac{x_1^2 - x_2^2}{\sqrt{\pi}} \right)$$

- The resulting Mercer feature map is

$$\Phi(\mathbf{x}) = \left(\sqrt{\frac{3}{2}}, \sqrt{2} x_1, \sqrt{2} x_2, \sqrt{2} x_1 x_2, \frac{x_1^2 - x_2^2}{\sqrt{2}} \right)$$

- Obviously, $\Phi(\mathbf{x})^\top \Phi(\mathbf{t}) = K(\mathbf{x}, \mathbf{t})$ for $\mathbf{x}, \mathbf{t} \in S^1$ *(exercice)*

RKHS of Mercer kernels

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

Reminder: expansion of Mercer kernel

Theorem

Denote by L_K the linear operator of $L^2_\nu(\mathcal{X})$ defined by:

$$\forall f \in L^2_\nu(\mathcal{X}), (L_K f)(\mathbf{x}) = \int K(\mathbf{x}, \mathbf{t}) f(\mathbf{t}) d\nu(\mathbf{t}).$$

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

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

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

RKHS construction

Theorem

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

$$\mathcal{H} = \left\{ f = \sum_{i=1}^{\infty} a_i \psi_i, \quad \text{with} \ \sum_{k=1}^{\infty} \frac{a_k^2}{\lambda_k} < \infty \right\}$$

endowed with the inner product:

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{k=1}^{\infty} \frac{a_k b_k}{\lambda_k}, \qquad \text{for } f = \sum_k a_k \psi_k, g = \sum_k b_k \psi_k .$$

Remark

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

Proof (1/6)

Sketch

In order to show that $\mathcal{H}$ 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 $\mathbf{x} \in \mathcal{X}$, $K_x \in \mathcal{H}$,
3. for any $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}$, $f(\mathbf{x}) = \langle f, K_x \rangle_{\mathcal{H}}$.

Proof (2/6)

$\mathcal{H}$ is a Hilbert space

Indeed the function:

$$\begin{aligned} L_K^{\frac{1}{2}} : & L^2_\nu(\mathcal{X}) \to \mathcal{H} \\ & \sum_{i=1}^{\infty} a_i \psi_i \mapsto \sum_{i=1}^{\infty} a_i \sqrt{\lambda_i} \psi_i \end{aligned}$$

is an isomorphism, therefore $\mathcal{H}$ is a Hilbert space, like $L^2_\nu(\mathcal{X})$. $\square$

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 $\mathbf{x} \in \mathcal{X}$, we have (if $f(x)$ makes sense):

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

Therefore convergence in $\| . \|_{\mathcal{H}}$ implies uniform convergence for functions.

Proof (4/6)

$\mathcal{H}$ is a space of continuous functions (cont.)

Let now $f_n = \sum_{i=1}^n a_i \psi_i \in \mathcal{H}$. The functions ψ_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 $f_c \in L^2_\nu(\mathcal{X})$ and

$$\| f_n - f_c \|_{L^2_\nu(\mathcal{X})} \underset{n \to \infty}{\to} 0.$$

On the other hand,

$$\| f - f_n \|_{L^2_\nu(\mathcal{X})} \leq \lambda_1 \| f - f_n \|_{\mathcal{H}} \underset{n \to \infty}{\to} 0,$$

therefore $f = f_c$. $\square$

Proof (5/6)

$K_x \in \mathcal{H}$

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

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

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

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

therefore $\varphi_x = K_x \in \mathcal{H}$. $\square$

Proof (6/6)

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

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

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

therefore:

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

which concludes the proof. $\square$

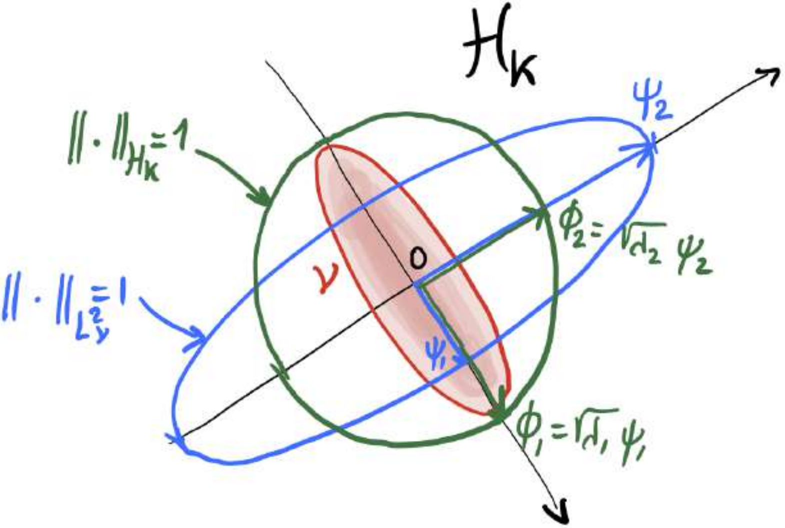
Remarks

- Although $\mathcal{H}$ was built from the eigenfunctions of L_K, which depend on the choice of the measure $d\nu(\mathbf{x})$, we know by uniqueness of the RKHS that $\mathcal{H}$ is independant of ν 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.

Summary


$\mathcal{H}_K$
$\|\cdot\|_{\mathcal{H}_K}=1$
$\|\cdot\|_{L^2_\nu}=1$
ψ_2
$\phi_2=\sqrt{\lambda_2}\,\psi_2$
0
ν
ψ_1
$\phi_1=\sqrt{\lambda_1}\,\psi_1$

Example: Sobolev space of periodic functions on $[0, 1]$

Corollary

For $\beta \in \mathbb{N}_*$, let the Mercer kernel with polynomially decaying eigenvalues:

$$\forall \mathbf{x}, \mathbf{t} \in [0,1]\,, \quad K(\mathbf{x}, \mathbf{t}) = \frac{1}{(2\beta)!} B_{2\beta}(\mathbf{x} - \mathbf{t} - \lfloor \mathbf{x} - \mathbf{t} \rfloor)\,,$$

where $B_{2\beta}$ is the (2β)-th Bernoulli polynomial. Then the RKHS is the set of functions $f : [0,1] \to \mathbb{R}$ 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

$$\| f \|_{\mathcal{H}}^2 = \pi^{-2\beta} \int_0^1 \left(f^{(\beta)}(\mathbf{x}) \right)^2 d\mathbf{x}\,.$$

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 ℓ_2 norm of the Fourier coefficients of $f^{(\beta)}$, which are (roughly) the Fourier coefficients of f multiplied by n^β. For details, see Tsybakov (2004, Proposition 1.14).

Outline

5. 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 betwwen $\mathcal{H}$ and $L^2_\nu(\mathcal{X})$

- We saw that

$$\begin{aligned} L_K^{\frac{1}{2}} : & L^2_\nu(\mathcal{X}) \to \mathcal{H} \\ & \sum_{i=1}^{\infty} a_i \psi_i \mapsto \sum_{i=1}^{\infty} a_i \sqrt{\lambda_i} \psi_i \end{aligned}$$

  is an isomorphism betwwen $\mathcal{H}$ and $L^2_\nu(\mathcal{X})$, i.e.,

$$\forall f \in L^2_\nu(\mathcal{X}) \,, \quad \| f \|_{L^2_\nu(\mathcal{X})} = \| L_K^{\frac{1}{2}} f \|_{\mathcal{H}} \,,$$

  and conversely,

$$\forall f \in \mathcal{H} \,, \quad \| f \|_{\mathcal{H}} = \| L_K^{-\frac{1}{2}} f \|_{L^2_\nu(\mathcal{X})} \,.$$

- 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 $(\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{X}^n$ and $(y_1, \ldots, y_n) \in \mathbb{R}^n$, KRR solves for any $\lambda > 0$:

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

- The solution is

$$\hat{f}_\lambda(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) , \text{ where } \boldsymbol{\alpha} = \left(\mathbf{K} + \lambda n \mathbf{I}\right)^{-1} \mathbf{y} .$$

Model

- Let K be a Mercer kernel over the compact set $\mathcal{X}$ and nondegenerate probability measure ν (i.e., $\nu(\mathcal{X}) = 1$). Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ be the eigenvalues of L_K, $\{\psi_i, i \geq 1\}$ the eigenvectors, and $\left\{\varphi_i = \sqrt{\lambda_i}\psi_i, i \geq 1\right\}$ 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

$$Y = f^*(X) + \epsilon \quad \text{where} \quad f^* \in \mathcal{H} \quad \text{and} \quad \epsilon \sim \mathcal{N}(0, \sigma^2)\,.$$

- We assume $(\mathbf{x}_i, y_i)_{i=1,\ldots,n}$ are i.i.d. realizations of (X, Y).
- We want to estimate the performance of KRR in terms of mean squared error:

$$MSE(\hat{f}_\lambda) = \mathbb{E}(Y - \hat{f}_\lambda(X))^2\,.$$

Decomposition of the MSE

Lemma

Let $\boldsymbol{\beta}^* \in \ell^2$ *such that* $f^* = \sum_{i \geq 1} \beta_i^* \varphi_i$, *let* Φ_N *the* $n \times \infty$ *matrix given by* $\Phi_n = (\varphi_j(\mathbf{x}_i))_{1 \leq i \leq n; 1 \leq j < +\infty}$. *and* $\mathcal{T} : \ell_2 \to \ell_2$ *be the diagonal operator* $\mathcal{T}(a_1, a_2, \dots) = (\lambda_1 a_1, \lambda_2 a_2, \dots)$.
Then it holds

$$MSE(\hat{f}_\lambda) - MSE(f^*) = B_\lambda + V_\lambda \,,$$

where

$$B_\lambda = \mathbb{E} \| \, \mathcal{T}^{\frac{1}{2}} \left(\mathbf{I} - \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I} \right)^{-1} \Phi_n^\top \Phi_n \right) \boldsymbol{\beta}^* \, \|_{\ell^2}^2 \,,$$

$$V_\lambda = \mathbb{E} \| \, \mathcal{T}^{\frac{1}{2}} \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I} \right)^{-1} \Phi_n^\top \boldsymbol{\varepsilon} \, \|_{\ell^2}^2 \,.$$

This corresponds to a classical decomposition of excess MSE as "bias + variance". Note that B_λ increases with λ, but V_λ decreases with λ.

Decomposition of the MSE: Proof (1/5)

- Since ϵ is independent of X and $\hat{f}_\lambda$, and $\mathbb{E}\epsilon = 0$ we have

$$\begin{aligned} MSE(\hat{f}_\lambda) &= \mathbb{E}\left(f^*(X) - \hat{f}_\lambda(X) + \epsilon\right)^2 \\ &= \mathbb{E}\left(f^*(X) - \hat{f}_\lambda(X)\right)^2 + \mathbb{E}\epsilon^2 \\ &= \mathbb{E}\| \, f^* - \hat{f}_\lambda \, \|^2_{L^2_\nu(\mathcal{X})} + MSE(f^*)\,. \end{aligned}$$

- Using the isometry between $L^2_\nu(\mathcal{X})$ and $\mathcal{H}$, we obtain

$$MSE(\hat{f}_\lambda) - MSE(f^*) = \mathbb{E}\| \, L_K^{\frac{1}{2}}(f^* - \hat{f}_\lambda) \, \|^2_{\mathcal{H}}\,.$$

Decomposition of the MSE: Proof (2/5)

- $\left\{\varphi_i = \sqrt{\lambda_i}\psi_i\right\}$, $i \geq 1$ is an ONB of $\mathcal{H}$, we can define the linear isomorphism:

$$\begin{aligned} e :&\mathcal{H} \to \ell^2 \\ &f = \sum_{i \geq 1} a_i \varphi_i \mapsto (a_1, a_2 \ldots)^\top \end{aligned}$$

- In other words,

$$e(f)_i = \langle f, \varphi_i \rangle_{\mathcal{H}} \ .$$

- In particular, for any $\mathbf{x} \in \mathcal{X}$,

$$e(K_\mathbf{x}) = (\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots)^\tau \ .$$

- In that base L_K is a diagonal operator $\mathcal{T} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots)$, i.e.,

$$\forall f = \sum_{i \geq 1} a_i \varphi_i \in \mathcal{H} \,, \quad e(L_K f) = \mathcal{T} e(f) = (\lambda_1 a_1, \lambda_2 a_2, \ldots)^\top \ .$$

Decomposition of the MSE: Proof (3/5)

- Let $\Phi_n = (e(K_{\mathbf{x}_1}), \ldots, e(K_{\mathbf{x}_n}))^\top$, i.e.,

$$\Phi_n = (\varphi_j(\mathbf{x}_i))_{1 \leq i \leq n; 1 \leq j < +\infty} .$$

- Then $\hat{f}_\lambda = \sum_{i=1}^n \alpha_i K_{\mathbf{x}_i}$ translates to

$$e(\hat{f}_\lambda) = \sum_{i=1}^n \alpha_i e\left(K_{\mathbf{x}_i}\right) = \Phi_n^\top \boldsymbol{\alpha} .$$

- Notice that

$$[\Phi_n \Phi_n^\top]_{ij} = \left\langle e(K_{\mathbf{x}_i}), e(K_{\mathbf{x}_j}) \right\rangle_{\ell^2} = \left\langle K_{\mathbf{x}_i}, K_{\mathbf{x}_j} \right\rangle_{\mathcal{H}} = K(\mathbf{x}_i, \mathbf{x}_j) ,$$

so $\Phi_n \Phi_n^\top = \mathbf{K}$ and $\boldsymbol{\alpha} = (\mathbf{K} + \lambda n \mathbf{I})^{-1} \mathbf{y}$ translates to

$$\boldsymbol{\alpha} = \left(\Phi_n \Phi_n^\top + \lambda n \mathbf{I}\right)^{-1} \mathbf{y} .$$

- Putting it all together, and using the matrix inversion lemma:

$$e(\hat{f}_\lambda) = \Phi_n^\top \left(\Phi_n \Phi_n^\top + \lambda n \mathbf{I}\right)^{-1} \mathbf{y} = \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I}\right)^{-1} \Phi_n^\top \mathbf{y} .$$

Decomposition of the MSE: Proof (4/5)

- Let $\boldsymbol{\beta}^* = (\beta_1^*, \beta_2^*, \ldots)^\top = e(f^*)$, i.e.,

  $$f^* = \sum_{i \geq 1} \beta_i^* \varphi_i \,.$$

  In particular, for any $\mathbf{x} \in \mathcal{X}$,

  $$f^*(\mathbf{x}) = \langle f^*, K_{\mathbf{x}} \rangle_{\mathcal{H}} = \langle \boldsymbol{\beta}^*, e(K_{\mathbf{x}}) \rangle_{\ell^2} \ .$$

- Then $y_i = f^*(\mathbf{x}_i) + \epsilon_i$ for $i = 1, \ldots, n$ translates to

  $$\mathbf{y} = \Phi_n \boldsymbol{\beta}^* + \boldsymbol{\varepsilon} \,,$$

  where $\boldsymbol{\varepsilon} = (\epsilon_1, \ldots, \epsilon_n)^\top$.

Decomposition of the MSE: Proof (5/5)

- This gives

$$
\begin{aligned}
e(f^* - \hat{f}_\lambda) &= \boldsymbol{\beta}^* - \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I}\right)^{-1} \Phi_n^\top (\Phi_n \boldsymbol{\beta}^* + \boldsymbol{\varepsilon}) \\
&= \left(\mathbf{I} - \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I}\right)^{-1} \Phi_n^\top \Phi_n\right) \boldsymbol{\beta}^* - \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I}\right)^{-1} \Phi_n^\top \boldsymbol{\varepsilon}\,,
\end{aligned}
$$

and therefore, since $\boldsymbol{\varepsilon}$ is independent of Φ_n:

$$
\begin{aligned}
\mathbb{E}\| L_K^{\frac{1}{2}}(f^* - \hat{f}_\lambda) \|_{\mathcal{H}}^2 &= \mathbb{E}\| e\left(L_K^{\frac{1}{2}}(f^* - \hat{f}_\lambda)\right) \|_{\ell^2}^2 \\
&= \mathbb{E}\| \mathcal{T}^{\frac{1}{2}} e\left(f^* - \hat{f}_\lambda)\right) \|_{\ell^2}^2 \\
&= \mathbb{E}\| \mathcal{T}^{\frac{1}{2}} \left(\mathbf{I} - \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I}\right)^{-1} \Phi_n^\top \Phi_n\right) \boldsymbol{\beta}^* \|_{\ell^2}^2 \\
&+ \mathbb{E}\| \mathcal{T}^{\frac{1}{2}} \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I}\right)^{-1} \Phi_n^\top \boldsymbol{\varepsilon} \|_{\ell^2}^2\,. \qquad \square
\end{aligned}
$$

Simplification

- B_λ and V_λ depend on the data through $\Phi_n \Phi_n^\top$, which is a random operator $\ell^2 \to \ell^2$.
- For "large n", we note that, for any $i, j \geq 1$:

$$\left[\Phi_n \Phi_n^\top\right]_{ij} = \sum_{k=1}^{n} \varphi_i(\mathbf{x}_k)\varphi_j(\mathbf{x}_k) \approx n \left\langle \varphi_i, \varphi_j \right\rangle_{L^2_\nu(\mathcal{X})} = n\sqrt{\mu_i \mu_j}\delta_{ij} \,,$$

  so

$$\Phi_n \Phi_n^\top \approx n\mathcal{T} \,.$$

- We now study B_λ and V_λ under the approximation "$\Phi_n \Phi_n^\top = n\mathcal{T}$" (and call $\tilde{B}_\lambda$ and $\tilde{V}_\lambda$ the corresponding approximations).
- The difference between B_λ and $\tilde{B}_\lambda$ (resp. V_λ 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

$$\tilde{V}_\lambda \leq \frac{\sigma^2}{n} \left[J + \frac{\sum_{i=J+1}^{+\infty} \lambda_i}{4\lambda} \right] .$$

The integer J (and λ) will be optimized later, depending on the assumptions we make on f^* and on the decrease of λ_i.

Proof: bias (1/2)

- Using $\mathcal{T} = \mathrm{diag}(\lambda_i; i \geq 1)$ and $\Phi_n \Phi_n^\top = n\mathcal{T}$, we get

  $$\mathcal{T}^{\frac{1}{2}} \left(\mathbf{I} - \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I} \right)^{-1} \Phi_n^\top \Phi_n \right) = \mathrm{diag} \left(\frac{\lambda \sqrt{\lambda_i}}{\lambda + \lambda_i}; i \geq 1 \right) ,$$

  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

  $$\sum_{i=1}^{J} \frac{\lambda^2 \lambda_i}{(\lambda + \lambda_i)^2} (\beta_i^*)^2 = \sum_{i=1}^{J} \frac{\lambda^2}{\lambda_i} \frac{\lambda_i^2}{(\lambda + \lambda_i)^2} (\beta_i^*)^2$$

  $$\leq \frac{\lambda^2}{\lambda_J} \sum_{i=1}^{J} (\beta_i^*)^2 \leq \frac{\lambda^2}{\lambda_J} \| \boldsymbol{\beta}^* \|_{\ell^2}^2 .$$

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

$$\sum_{i \geq J+1}^{\infty} \frac{\lambda^2 \lambda_i}{(\lambda + \lambda_i)^2} (\beta_i^*)^2 \leq \lambda_{J+1} \sum_{i \geq J+1}^{\infty} (\beta_i^*)^2 \leq \lambda_{J+1} \| \boldsymbol{\beta}^* \|_{\ell^2}^2 .$$

- Noting that $\| \boldsymbol{\beta}^* \|_{\ell^2} = \| f^* \|_{\mathcal{H}}$, we finally get

$$\tilde{B}_\lambda \leq \left(\frac{\lambda^2}{\lambda_J} + \lambda_{J+1} \right) \| f^* \|_{\mathcal{H}}^2 . \qquad \square$$

Proof: variance (1/2)

Using $\mathcal{T} = \operatorname{diag}(\lambda_i; i \geq 1)$, $\Phi_n \Phi_n^\top = n\mathcal{T}$ and $\mathbb{E}\varepsilon\varepsilon^\top = \sigma^2 \mathbf{I}$, we get

$$\begin{aligned}
\tilde{V}_\lambda &= \mathbb{E}\| \mathcal{T}^{\frac{1}{2}} \left(\Phi_n^\top \Phi_n + \lambda n \mathbf{I}\right)^{-1} \Phi_n^\top \varepsilon \|_{\ell^2}^2 \\
&= \frac{1}{n^2} \mathbb{E}\| \mathcal{T}^{\frac{1}{2}} (\mathcal{T} + \lambda \mathbf{I})^{-1} \Phi_n^\top \varepsilon \|_{\ell^2}^2 \\
&= \frac{1}{n^2} \mathbb{E}\mathrm{Trace}\left[\mathcal{T}^{\frac{1}{2}} (\mathcal{T} + \lambda \mathbf{I})^{-1} \Phi_n^\top \varepsilon\varepsilon^\top \Phi_n (\mathcal{T} + \lambda \mathbf{I})^{-1} \mathcal{T}^{\frac{1}{2}}\right] \\
&= \frac{1}{n^2} \mathrm{Trace}\left[\mathcal{T}^{\frac{1}{2}} (\mathcal{T} + \lambda \mathbf{I})^{-1} \Phi_n^\top \mathbb{E}\left(\varepsilon\varepsilon^\top\right) \Phi_n (\mathcal{T} + \lambda \mathbf{I})^{-1} \mathcal{T}^{\frac{1}{2}}\right] \\
&= \frac{\sigma^2}{n} \mathrm{Trace}\left[\mathcal{T}^{\frac{1}{2}} (\mathcal{T} + \lambda \mathbf{I})^{-1} \mathcal{T} (\mathcal{T} + \lambda \mathbf{I})^{-1} \mathcal{T}^{\frac{1}{2}}\right] \\
&= \frac{\sigma^2}{n} \left[\sum_{i=1}^{J} \frac{\lambda_i^2}{(\lambda_i + \lambda)^2} + \sum_{i=J+1}^{+\infty} \frac{\lambda_i^2}{(\lambda_i + \lambda)^2}\right].
\end{aligned}$$

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 $t \rightarrow \frac{t}{(t+\lambda)^2}$ reaches its maximum at $t = \lambda$ equal to $\frac{1}{4\lambda}$, therefore

$$\sum_{i=J+1}^{+\infty} \frac{\lambda_i^2}{(\lambda_i + \lambda)^2} \leq \frac{\sum_{i=J+1}^{+\infty} \lambda_i}{4\lambda} \, .$$

- Combining both terms finally gives

$$\tilde{V}_\lambda \leq \frac{\sigma^2}{n} \left[J + \frac{\sum_{i=J+1}^{+\infty} \lambda_i}{4\lambda} \right] . \qquad \square$$

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
$$\tilde{B}_\lambda + \tilde{V}_\lambda \leq O\left\{\left(\| f^* \|_{\mathcal{H}}^2 + \sigma^2\right) n^{-\frac{s}{s+1}}\right\}.$$
- **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
$$\tilde{B}_\lambda + \tilde{V}_\lambda \leq O\left\{\left(\| f^* \|_{\mathcal{H}}^2 + \sigma^2\right) \frac{\log(n)}{n}\right\}.$$
- **Finite rank kernels.** Suppose there is $J \geq 1$ such that $\lambda_J = \lambda_{J+1} = \ldots = 0$. Let $\lambda = n^{-1}$. Then
$$\tilde{B}_\lambda + \tilde{V}_\lambda \leq O\left\{\left(\| f^* \|_{\mathcal{H}}^2 + \sigma^2\right) \frac{J}{n}\right\}.$$

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 $d\nu(\mathbf{x})$.
- The rates are minimax optimal (Caponnetto and De Vito, 2007).
- In particular, for polynomial-decay kernels, $B_{\mathcal{H}}(r) \subset L^2_\nu(\mathcal{X})$ is a Sobolev space of $q-1$ times absolutely continuous and differentiable functions f with $\| f^q \|_{L^2_\nu(\mathcal{X})} < +\infty$, 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

$$J \leq C^{\frac{1}{s}} \lambda^{-\frac{1}{s}} = C^{\frac{1}{s}} n^{\frac{1}{s+1}} \,.$$

Proof for polynomial-decay kernels (2/3)

- To upper bound the sum, let $J_0 = \lfloor C^{\frac{1}{s}} n^{\frac{1}{s+1}} \rfloor + 1$. Then:

$$\begin{aligned}\sum_{i=J+1}^{+\infty} \lambda_i &= \sum_{i=J+1}^{J_0} \lambda_i + \sum_{i=J_0+1}^{+\infty} \lambda_i \\ &\leq J_0 \lambda + C \int_{J_0}^{+\infty} t^{-s} dt \\ &\leq J_0 n^{-\frac{s}{s+1}} + \frac{C}{s-1} J_0^{1-s} .\end{aligned}$$

- 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

$$\sum_{i=J+1}^{+\infty} \lambda_i \leq \left(\frac{s}{s-1} C^{\frac{1}{s}} + 1 \right) n^{\frac{1-s}{s+1}} .$$

- Finally,

$$\begin{aligned} \tilde{V}_\lambda &= \frac{\sigma^2}{n} \left[J + \frac{\sum_{i=J+1}^{+\infty} \lambda_i}{4} n^{-\frac{s}{s+1}} \right] \\ &\leq \frac{\sigma^2}{n} \left[C^{\frac{1}{s}} n^{\frac{1}{s+1}} + \frac{1}{4} \left(\frac{s}{s-1} C^{\frac{1}{s}} + 1 \right) n^{\frac{1-s}{s+1}} n^{-\frac{s}{s+1}} \right] \\ &\leq \sigma^2 \left[C^{\frac{1}{s}} \left(1 + \frac{s}{4(s-1)} \right) + \frac{1}{4} \right] n^{\frac{-s}{s+1}} . \qquad \square \end{aligned}$$

Proof sketch for exponential-decay kernels

- We proceed similarly.
- From $\lambda \leq \lambda_J$ we deduce $J \leq O(\log(n))$.
- Using $J_0 = \lfloor \alpha^{-1} \log(n) \rfloor + 1$ we deduce $\sum_{i \geq J+1} \lambda_i \leq 0 \left(\frac{\log(n)^2}{n} \right)$.
- *Details left as exercice; see Dicker et al. (2015, corollary 2).* $\square$

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. $\square$

Outline

5 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:

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

  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 $K(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x} - \mathbf{y})$.
- Harmonic analysis also bring kernels well beyond vector spaces, e.g., groups and semigroups

Translation invariant kernels on $\mathbb{Z}$

Definition

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

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{Z}\,, \quad K\left(\mathbf{x}, \mathbf{y}\right) = a_{\mathbf{x}-\mathbf{y}}$$

for some sequence $\left\{a_n\right\}_{n \in \mathbb{Z}}$. 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.:

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{Z}\,, \quad K\left(\mathbf{x}, \mathbf{y}\right) = a_{\mathbf{x}-\mathbf{y}}$$

for some sequence $\left\{a_n\right\}_{n\in\mathbb{Z}}$. Such a sequence is called positive definite if the corresponding kernel K is p.d.

Theorem (Herglotz)

A sequence $\left\{a_n\right\}_{n\in\mathbb{Z}}$ 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 2π identified.

Fourier-Stieltjes transform on the torus

- Let $\mathbb{T}$ the torus $[0, 2\pi]$ with 0 and 2π 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 $\psi(f) = \frac{1}{2\pi} \int_{\mathbb{T}} f(t)\overline{d\mu(t)}$ (Riesz theorem).

Definition (Fourier-Stieltjes coefficients)

For any $\mu \in M(\mathbb{T})$, the Fourier-Stieltjes coefficients of μ 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:

  $$\mu = dt\,, \quad a_n = \hat{\mu}(n) = \frac{1}{2\pi}\int_{\mathbb{T}} e^{int}\,dt = \begin{cases} 1 & \text{if } n = 0\,, \\ 0 & \text{otherwise.} \end{cases}$$

  The resulting kernel is $K(\mathbf{x}, \mathbf{t}) = \mathbf{1}(\mathbf{x} = \mathbf{t})$.

- Constant kernel: for $C \geq 0$,

  $$\mu = 2\pi C \delta_0\,, \quad a_n = \hat{\mu}(n) = C \int_{\mathbb{T}} e^{int}\delta_0(t) = C\,,$$

  resulting in $K(\mathbf{x}, \mathbf{t}) = C$

Proof of Herglotz's theorem: $\Leftarrow$

If $a_n = \hat{\mu}(n)$ for $\mu \in M(\mathbb{T})$ positive, then for any $n \in \mathbb{N}$, $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{Z}$ and $z_1, \dots, z_n \in \mathbb{R}$ (or $\mathbb{C}$) :

$$\begin{aligned}
\sum_{i=1}^{n} \sum_{j=1}^{n} z_i \bar{z}_j a_{\mathbf{x}_i - \mathbf{x}_j} &= \frac{1}{2\pi} \sum_{i=1}^{n} \sum_{j=1}^{n} z_i \bar{z}_j \int_{\mathbb{T}} e^{-i(\mathbf{x}_i - \mathbf{x}_j)t} d\mu(t) \\
&= \frac{1}{2\pi} \sum_{i=1}^{n} \sum_{j=1}^{n} z_i \bar{z}_j \int_{\mathbb{T}} e^{-i\mathbf{x}_i t} e^{i\mathbf{x}_j t} d\mu(t) \\
&= \frac{1}{2\pi} \int_{\mathbb{T}} | \sum_{j=1}^{n} z_j e^{-i\mathbf{x}_j t} |^2 d\mu(t) \\
&\geq 0.
\end{aligned}$$

$\square$

Proof of Herglotz's theorem: $\Rightarrow$ (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:

$$\begin{aligned} 0 \leq \sum_{k=-N}^{N} \sum_{l=-N}^{N} a_{k-l} z_k \bar{z}_l &= \sum_{k=-N}^{N} \sum_{l=-N}^{N} a_{k-l} e^{i(k-l)t} \\ &= \sum_{k=-2N}^{2N} (2N+1-|k|) a_k e^{ikt} \\ &= (2N+1) \underbrace{\sum_{k\in\mathbb{Z}} \max\left(0, 1 - \frac{|k|}{2N+1}\right) a_k e^{ikt}}_{\sigma_{2N}(t)} \end{aligned}$$

Proof of Herglotz's theorem: $\Rightarrow$ (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

$$\begin{aligned} \| \mu_N \|_{M(\mathbb{T})} &= \sup_{\| f \|_\infty \leq 1} \int_{\mathbb{T}} f(t)\sigma_N(t)dt \\ &= \int_{\mathbb{T}} \sigma_N(t)dt \qquad \text{(take } f = 1 \text{ because } \sigma_N(t) \geq 0) \\ &= \sum_{n=-N}^{N} \int_{\mathbb{T}} a_n \left(1 - \frac{|n|}{N+1}\right) e^{int} dt \\ &= a_0 \end{aligned}$$

Proof of Herglotz's theorem: $\Rightarrow$ (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

$$
\begin{aligned}
&\lim_{N\to+\infty} \int_{\mathbb{T}} P(t) d\mu_N(t) \\
&\qquad = \lim_{N\to+\infty} \sum_{k=-K}^{K} \sum_{n=-N}^{N} \int_{\mathbb{T}} a_n b_k \left(1 - \frac{|n|}{N+1}\right) e^{i(n-k)t} dt \\
&\qquad = \sum_{k=-K}^{K} a_k b_k \lim_{N\to+\infty} \left(1 - \frac{|n|}{N+1}\right) \\
&\qquad = \sum_{k=-K}^{K} a_k b_k \\
&\qquad = \sum_{k\in\mathbb{Z}} a_k \hat{P}(k)
\end{aligned}
$$

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 μ is a positive measure because if $f \geq 0$

$$\int_{\mathbb{T}} f(t) d\mu(t) = \Psi(f) = \lim_{n\to+\infty} \Psi(P_n) = \lim_{n,k\to+\infty} \Psi_k(P_n) \geq 0 \qquad \square$$

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.:

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d \,, \quad K\left(\mathbf{x}, \mathbf{y}\right) = \varphi\left(\mathbf{x} - \mathbf{y}\right)$$

for some function $\varphi : \mathbb{R}^d \to \mathbb{R}$. Such a function φ 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.:

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d\,, \quad K\left(\mathbf{x}, \mathbf{y}\right) = \varphi\left(\mathbf{x} - \mathbf{y}\right)$$

for some function $\varphi : \mathbb{R}^d \to \mathbb{R}$. Such a function φ 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 $\left(C_0(\mathbb{R}^d)\right)^*$: 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 μ is the function:

$$\forall \boldsymbol{\omega} \in \mathbb{R}^d \,, \quad \hat{\mu}(\boldsymbol{\omega}) = \int_{\mathbb{R}^d} e^{-i\boldsymbol{\omega}^\top \mathbf{x}} d\mu(\mathbf{x})$$

Fourier-Stieltjes transform on $\mathbb{R}^d$

- This extends the standard Fourier transform for integrable functions by taking $d\mu(\mathbf{x}) = f(\mathbf{x})d\mathbf{x}$.
- For $\mu \in M(\mathbb{R}^d)$, $\hat{\mu}$ is still uniformly continuous, but $\hat{\mu}(\boldsymbol{\omega})$ does not necessarily go to 0 at infinity (e.g., take the Dirac $\mu = \delta_0$, then $\hat{\mu}(\boldsymbol{\omega}) = 1$ for all $\boldsymbol{\omega}$)
- Parseval's formula: if $\mu \in M(\mathbb{R}^d)$, and both $g, \hat{g}$ are in $L^1(\mathbb{R}^d)$, then

$$\int_{\mathbb{R}^d} g(\mathbf{x})d\mu(\mathbf{x}) = \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} \hat{g}(\boldsymbol{\omega})\hat{\mu}(-\boldsymbol{\omega})d\boldsymbol{\omega}\,.$$

  In particular, if $g \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$,

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

Proof of Bochner's theorem: $\Leftarrow$

If $\varphi = \hat{\mu}$ for some $\mu \in M(\mathbb{T})$ positive, then for any $n \in \mathbb{N}$, $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ and $z_1, \dots, z_n \in \mathbb{R}$ (or $\mathbb{C}$) :

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{n} z_i \bar{z}_j \varphi\left(\mathbf{x}_i - \mathbf{x}_j\right) &= \sum_{i=1}^{n}\sum_{j=1}^{n} z_i \bar{z}_j \int_{\mathbb{R}^d} e^{-i(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{t}} d\mu(\mathbf{t}) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} z_i \bar{z}_j \int_{\mathbb{R}^d} e^{-i\mathbf{x}_i^\top \mathbf{t}} e^{i\mathbf{x}_j^\top \mathbf{t}} d\mu(\mathbf{t}) \\
&= \int_{\mathbb{R}^d} |\sum_{j=1}^{n} z_j e^{-i\mathbf{x}_j^\top \mathbf{t}}|^2 d\mu(\mathbf{t}) \\
&\geq 0.
\end{aligned}$$

If μ is symmetric then, in addition, φ is real-valued. □

Proof of Bochner's theorem: $\Rightarrow$ (1/5)

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 φ 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 $\varphi = \hat{\mu}$. $\square$

Note: the converse is also true.

Proof of Bochner's theorem: $\Rightarrow$ (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 λ).

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 λ to ensure
$$\sup_{t \in \mathbb{T}} |\, G_\lambda(t) \,| \leq \sup_{x \in \mathbb{R}} |\, g(x) \,| + \epsilon$$

Proof of Bochner's theorem: ⇒ (3/5)

- In addition, for any $n \in \mathbb{Z}$:

$$\begin{aligned}
\hat{G}_\lambda(n) &= \frac{1}{2\pi} \int_{\mathbb{T}} e^{-int} G_\lambda(t) dt \\
&= \frac{1}{2\pi} \sum_{m \in \mathbb{Z}} \int_0^{2\pi} e^{-int} g\left(\frac{t + 2\pi m}{\lambda}\right) dt \\
&= \frac{\lambda}{2\pi} \sum_{m \in \mathbb{Z}} \int_{\frac{2\pi m}{\lambda}}^{\frac{2\pi(m+1)}{\lambda}} e^{-in(\lambda u + 2\pi m)} g(u) du \\
&= \frac{\lambda}{2\pi} \sum_{m \in \mathbb{Z}} \int_{\frac{2\pi m}{\lambda}}^{\frac{2\pi(m+1)}{\lambda}} e^{-in\lambda u} g(u) du \\
&= \frac{\lambda}{2\pi} \int_{\mathbb{R}} e^{-in\lambda u} g(u) du \\
&= \frac{\lambda}{2\pi} \hat{g}(\lambda n)
\end{aligned}$$

Proof of Bochner's theorem: $\Rightarrow$ (4/5)

- This gives:

$$\begin{aligned}
\left| \frac{\lambda}{2\pi} \sum_{n\in\mathbb{Z}} \hat{g}(\lambda n)\varphi(-\lambda n) \right| &= \left| \sum_{n\in\mathbb{Z}} \hat{G}_\lambda(n)\overline{\hat{\mu}_\lambda(-n)} \right| \\
&= \left| \frac{1}{2\pi} \int_{\mathbb{T}} G_\lambda(t)\overline{d\mu_\lambda(t)} \right| \qquad \text{(Parceval)} \\
&\leq \| \mu_\lambda \|_{M(\mathbb{T})} \sup_{t\in\mathbb{T}} | G_\lambda(t) | \\
&\leq C \sup_{t\in\mathbb{T}} | G_\lambda(t) | \\
&\leq C \sup_{x\in\mathbb{R}} | g(x) | + C\epsilon
\end{aligned}$$

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 $\frac{1}{2\pi}\int_{\mathbb{T}} G_\lambda(t)\overline{d\mu_\lambda(t)}$ for small λ, which is ≥ 0 because μ_λ is a positive measure and $G_\lambda \geq 0$ like g. Consequently, μ is a positive measure. $\square$

RKHS of translation invariant kernels

Theorem

Let $K(\mathbf{x}, \mathbf{t}) = \varphi(\mathbf{x} - \mathbf{t})$ *be a translation invariant p.d. kernel, such that* φ *is integrable on* $\mathbb{R}^d$ *as well as its Fourier transform* $\hat{\varphi}$*. The subset* $\mathcal{H}$ *of* $L_2\left(\mathbb{R}^d\right)$ *that consists of integrable and continuous functions* f *such that:*

$$\| f \|_{\mathcal{H}}^2 := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \frac{\left| \hat{f}(\omega) \right|^2}{\hat{\varphi}(\omega)} d\omega < +\infty \,,$$

endowed with the inner product:

$$\langle f, g \rangle_{\mathcal{H}} := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \frac{\hat{f}(\omega)\overline{\hat{g}\,(\omega)}}{\hat{\varphi}(\omega)} d\omega$$

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

Proof

- $\mathcal{H}$ is a Hilbert space: exercise.
- For $\mathbf{x} \in \mathbb{R}^d$, $K_\mathbf{x}(\mathbf{y}) = K(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x} - \mathbf{y})$ therefore:

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

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

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

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

$$\begin{aligned} \langle f, K_\mathbf{x} \rangle_\mathcal{H} &= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \frac{\hat{K}_\mathbf{x}(\omega) \overline{\hat{f}(\omega)}}{\hat{\varphi}(\omega)} d\omega = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \overline{\hat{f}(\omega)} e^{-i\omega^\top \mathbf{x}} d\omega \\ &= f(\mathbf{x}) \end{aligned}$$

☐

Example

Gaussian kernel

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

corresponds to:

$$\varphi(t)=e^{-\frac{t^2}{2\sigma^2}}$$

$$\hat{\varphi}\left(\omega\right)=e^{-\frac{\sigma^2\omega^2}{2}}$$

and

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

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

Example

Laplace kernel

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

corresponds to:

$$\varphi(to)=\frac{1}{2}e^{-\gamma\left|\,t\,\right|}$$

$$\hat{\varphi}\left(\omega\right)=\frac{\gamma}{\gamma^2+\omega^2}$$

and

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

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

Example

Low-frequency filter

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

corresponds to:

$$\varphi(t)=\frac{\sin\left(\Omega t\right)}{\pi t}$$
$$\hat{\varphi}\left(\omega\right)=1_{[-\Omega,\Omega]}(\omega)$$

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,

	Kernel	RKHS $\mathcal{H}$
Green	Green func. of D^*D	$L_2(\mathcal{X})$ with $\langle Df, Dg \rangle_{L_2(\mathcal{X})}$
Mercer	$\sum_{k=1}^{\infty} \lambda_k \psi_k(x)\psi_k(y)$	$\left\{ f = \sum_{k=1}^{\infty} a_k \psi_k : \sum_{k=1}^{\infty} \frac{a_k^2}{\lambda_k} < +\infty \right\}$
Fourier	$\kappa(x-y) \propto \int \hat{\kappa}(\omega) e^{i\omega(x-y)} d\omega$	$\left\{ f \in \underbrace{L_2(\mathbb{R}^d)}_{\substack{+continuous \\ +integrable}} : \int \frac{\lvert \hat{f}(\omega) \rvert^2}{\hat{\kappa}(\omega)} d\omega < +\infty \right\}$

Recap on Green, Mercer, Bochner families

Up to specific assumptions for each of the following kernel families,

	Kernel	Squared Norm $\|.\|_{\mathcal{H}}^2$
Green	Green func. of $D^* D$	$\|Df\|_{L_2(\mathcal{X})}^2$
Mercer	$\sum_{k=1}^{\infty} \lambda_k \psi_k(x) \psi_k(y)$	$\sum_{k=1}^{\infty} \frac{a_k^2}{\lambda_k}$ for $f = \sum_{k=1}^{\infty} a_k \psi_k$
Fourier	$\kappa(x - y)$	$\frac{1}{(2\pi)^d} \int \frac{\|\hat{f}(\omega)\|^2}{\hat{\kappa}(\omega)} d\omega$

Outline

5 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 $* : S \to S$ 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:

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

is a p.d. kernel on S.

Example: translation invariant kernels

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

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

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

Semicharacters

Definition

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

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

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

Remarks

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

Semicharacters are p.d.

Lemma

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

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

Proof

Direct from definition, e.g.,

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

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) $\alpha\left(s^{*}\right) = \alpha(s)$.
- A function $f : S \to \mathbb{R}$ is called exponentially bounded if there exists an absolute value α and a constant $C > 0$ s.t. $\mid f(s) \mid \leq C\alpha(s)$ for $s \in S$.

Theorem

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

Remarks

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

Example 1: $(R_+, +, Id)$

Semicharacters

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

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

  for $a \in [0, +\infty]$ (left as exercice).
- Non-bounded semicharacters are more difficult to characterize; in fact there exist nonmeasurable solutions of the equation $h(x+y) = h(x)h(y)$.

Example 1: $(R_+, +, Id)$ (cont.)

P.d. functions

- By the integral representation theorem for bounded semi-characters we obtain that a function $\varphi : \mathbb{R}_+ \to \mathbb{R}$ 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 $\mu \in \mathcal{M}_+^b\left(\mathbb{R}_+\right)$ and $b \geq 0$.
- The first term is the Laplace transform of μ. φ is p.d., bounded and continuous iff it is the Laplace transform of a measure in $\mathcal{M}_+^b\left(\mathbb{R}\right)$.

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

Setting

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

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

Semigroup of bounded measures

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

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

  where a_i is the number of occurrences on $\mathbf{x}_i$ 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 $\left(\mathcal{M}_+^b(\mathcal{U}), +, Id\right)$ of bounded Radon measures on $\mathcal{U}$ and to search for p.d. functions φ on this semigroup.

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

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 $\left(\mathcal{M}_+^b(\mathcal{U}), +\right)$.
- Conversely, ρ 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.

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

Corollary

Let $\mathcal{U}$ be a Hausdorff space. For any Radon measure $\mu \in \mathcal{M}_+^c\left(C\left(\mathcal{U}\right)\right)$ 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 $\mathcal{M}_+^b\left(\mathcal{U}\right)$ (endowed with the topology of weak convergence):

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

Remarks

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

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

Example : entropy kernel

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

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

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

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

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

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

Examples : inverse generalized variance kernel

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

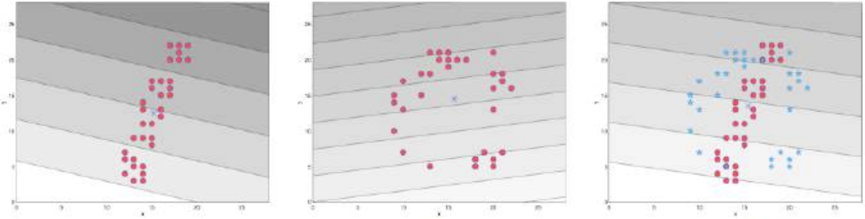
$$\Sigma(\mu) = \mu\left[xx^\top\right] - \mu[x]\,\mu[x]^\top .$$

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

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

- Generalization possible with regularization and kernel trick.

Application of semigroup kernel



$\Sigma_{1,1} = 0.0552$
$\Sigma_{2,2} = 0.0013$

$\Sigma'_{1,1} = 0.0441$
$\Sigma'_{2,2} = 0.0237$

$\Sigma''_{1,1} = 0.0497$
$\Sigma''_{2,2} = 0.0139$

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

Kernelization of the IGV kernel

Motivations

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

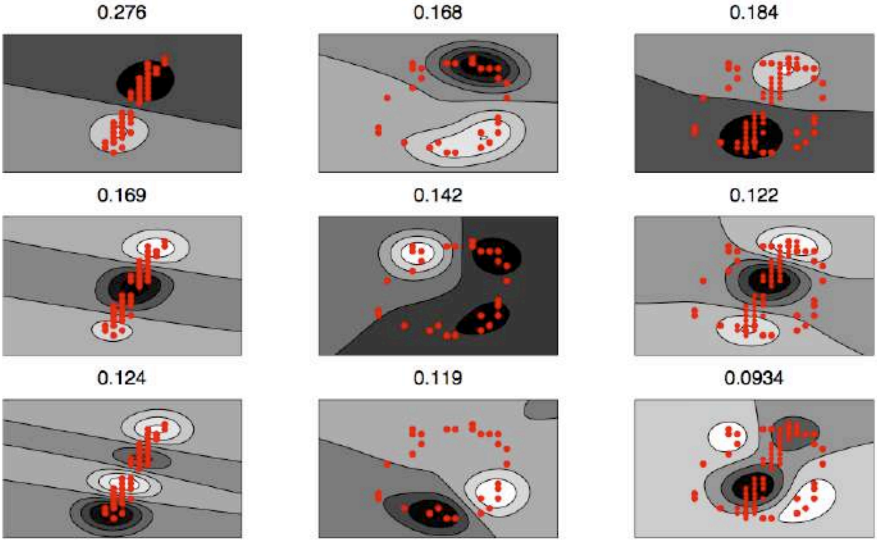
Solution

1. Regularization:

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

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

Illustration of kernel IGV kernel



Semigroup kernel remarks

Motivations

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

Kernel examples: Summary

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

Outline

1. Kernels and RKHS

2. Kernel tricks

3. Kernel Methods: Supervised Learning

4. Kernel Methods: Unsupervised 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

5. 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 $\mathbf{x}$, compute the Fisher score vector:

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

  which can be interpreted as the local contribution of each parameter.
- Form the kernel (Jaakkola et al., 2000):

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

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

Note: when θ_0 is the ML estimator, $\mathbb{E}[\Phi_{\theta_0}(\mathbf{x})] = 0$ and $\mathbf{I}(\theta_0)$ is a covariance matrix.

Fisher kernel properties (1/2)

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

Fisher kernel properties (2/2)

Lemma

The Fisher kernel is invariant under change of parametrization.

- Consider indeed 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 $\mathbf{I}(\lambda_0)^{-1} = \mathbf{J}^{-1}\mathbf{I}(\theta_0)^{-1}\mathbf{J}^{\top -1}$:

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \Phi_{\theta_0}(\mathbf{x})^\top \mathbf{I}(\theta_0)^{-1}\Phi_{\theta_0}(\mathbf{x}') = \Phi_{\lambda_0}(\mathbf{x})^\top \mathbf{I}(\lambda_0)^{-1}\Phi_{\lambda_0}(\mathbf{x}').$$

Fisher kernel in practice

- $\Phi_{\theta_0}(\mathbf{x})$ can be computed explicitly for many models (e.g., HMMs), where the model is first estimated from data.
- $\mathbf{I}(\theta_0)$ is often replaced by the identity matrix for simplicity.
- Several different models (i.e., different θ_0) can be trained and combined.
- The Fisher vectors are defined as $\varphi_{\theta_0}(\mathbf{x}) = \mathbf{I}(\theta_0)^{-1/2}\Phi_{\theta_0}(\mathbf{x})$. They are explicitly computed and correspond to an explicit embedding: $K(\mathbf{x},\mathbf{x}') = \varphi_{\theta_0}(\mathbf{x})^\top \varphi_{\theta_0}(\mathbf{x}')$.

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

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

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

and thus

$$\frac{\partial \log P_\theta(x)}{\partial \mu} = \alpha(x - \mu), \qquad \frac{\partial \log P_\theta(x)}{\partial \alpha} = \frac{1}{2} \left[\frac{1}{\alpha} - (x - \mu)^2 \right],$$

and (exercise)

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

The Fisher vector is then

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

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

Now consider an i.i.d. data model over a set of data points $x_1, \ldots, x_n$ 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.

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

- where

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \hat{\sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2.$$

Application: Aggregation of visual words (1/5)

- **Patch extraction and description stage:** In various contexts, images may be described as a set of patches $\mathbf{x}_1, \ldots, \mathbf{x}_n$ computed at interest points. For example, SIFT, HOG, LBP, color histograms, convolutional features...
- **Coding stage:** The set of patches is then encoded into a single representation $\varphi(\mathbf{x}_i)$, typically in a high-dimensional space.
- **Pooling stage:** For example, sum pooling

$$\varphi(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} \varphi(\mathbf{x}_i).$$

Fisher vectors with a Gaussian Mixture Model (GMM) is a simple and effective aggregation technique (Perronnin and Dance, 2007).

Application: Aggregation of visual words (2/5)

Let $\theta = (\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)_{j=1\ldots,k}$ be the parameters of a GMM with k Gaussian components. Then, the probabilistic model is given by

$$P_\theta(\mathbf{x}) = \sum_{j=1}^{k} \pi_j \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j).$$

Remarks

- Each mixture component corresponds to a visual word, with a mean, variance, and mixing weight.
- Diagonal covariances $\boldsymbol{\Sigma}_j = \mathrm{diag}\left(\sigma_{j1}, \ldots, \sigma_{jp}\right) = \mathrm{diag}\left(\boldsymbol{\sigma}_j\right)$ 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 $\varphi_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_n) =$

$$[\varphi_{\pi_1}(\mathbf{X}), \ldots, \varphi_{\pi_p}(\mathbf{X}), \varphi_{\boldsymbol{\mu}_1}(\mathbf{X})^\top, \ldots, \varphi_{\boldsymbol{\mu}_p}(\mathbf{X})^\top, \varphi_{\boldsymbol{\sigma}_1}(\mathbf{X})^\top, \ldots, \varphi_{\boldsymbol{\sigma}_p}(\mathbf{X})^\top]^\top,$$

with

$$\varphi_{\boldsymbol{\mu}_j}(\mathbf{X}) = \frac{1}{n\sqrt{\pi_j}} \sum_{i=1}^{n} \gamma_{ij}(\mathbf{x}_i - \boldsymbol{\mu}_j)/\boldsymbol{\sigma}_j$$

$$\varphi_{\boldsymbol{\sigma}_j}(\mathbf{X}) = \frac{1}{n\sqrt{2\pi_j}} \sum_{i=1}^{n} \gamma_{ij} \left[(\mathbf{x}_i - \boldsymbol{\mu}_j)^2/\boldsymbol{\sigma}_j^2 - 1\right],$$

where, with an abuse of notation, the division between two vectors is meant elementwise and the scalars γ_{ij} can be interpreted as the soft-assignment of word i to component j:

$$\gamma_{ij} = \frac{\pi_j \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\sigma}_j)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_l, \boldsymbol{\sigma}_l)}.$$

Application: Aggregation of visual words (4/5)

Finally, we also have the following interpretation of encoding first and second-order statistics:

$$\varphi_{\boldsymbol{\mu}_j}(\mathbf{X}) = \frac{\gamma_j}{\sqrt{\pi_j}}(\hat{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_j)/\boldsymbol{\sigma}_j$$
$$\varphi_{\boldsymbol{\sigma}_j}(\mathbf{X}) = \frac{\gamma_j}{\sqrt{2\pi_j}}(\hat{\boldsymbol{\sigma}}_j^2 - \boldsymbol{\sigma}_j^2)/\boldsymbol{\sigma}_j^2,$$

with

$$\gamma_j = \sum_{i=1}^{n} \gamma_{ij} \quad \text{and} \quad \hat{\boldsymbol{\mu}}_j = \frac{1}{\gamma_j}\sum_{i=1}^{n} \gamma_{ij}\mathbf{x}_i \quad \text{and} \quad \hat{\boldsymbol{\sigma}}_j = \frac{1}{\gamma_j}\sum_{i=1}^{n} \gamma_{ij}(\mathbf{x}_i - \boldsymbol{\mu}_j)^2.$$

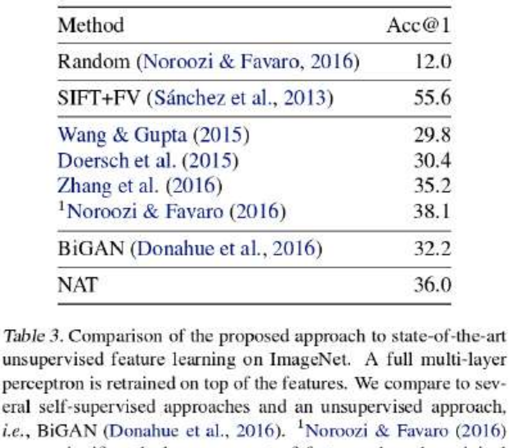
The component $\varphi_{\boldsymbol{\pi}}(\mathbf{X})$ is often dropped due to its negligible contribution in practice, and the resulting representation is of dimension $2kp$ where p is the dimension of the $\mathbf{x}_i$'s.

Application: Aggregation of visual words (5/5)

- FVs were state-of-the-art image representations before the revival of convolutional neural networks in 2012.

Application: Aggregation of visual words (5/5)

- 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.



Method	Acc@1
Random (Noroozi & Favaro, 2016)	12.0
SIFT+FV (Sánchez et al., 2013)	55.6
Wang & Gupta (2015)	29.8
Doersch et al. (2015)	30.4
Zhang et al. (2016)	35.2
[1]Noroozi & Favaro (2016)	38.1
BiGAN (Donahue et al., 2016)	32.2
NAT	36.0

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). [1]Noroozi & Favaro (2016)

Relation to classification with generative models (1/3)

Assume that we have a generative probabilistic model P_θ 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 θ, 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:

$$\hat{y}(x) = \underset{k=1,\ldots,p}{\mathrm{argmax}}\ P_\theta(Y = k|x),$$

where $P_\theta(Y = k|x)$ is short for $P_\theta(Y = k|X = x)$ and

$$\begin{aligned} P_\theta(Y = k|x) &= P_\theta(x|Y = k)P_\theta(Y = k)/P_\theta(x) \\ &= P_\theta(x|Y = k)\pi_k / \sum_{k'=1}^{p} P_\theta(x|Y = k')\pi_{k'} \end{aligned}$$

Relation to classification with generative models (2/3)

Then, consider the Fisher score

$$\begin{aligned}\nabla_\theta \log P_\theta(x) &= \frac{1}{P_\theta(x)} \nabla_\theta P_\theta(x) \\ &= \frac{1}{P_\theta(x)} \nabla_\theta \sum_{k=1}^{p} P_\theta(x, Y = k) \\ &= \frac{1}{P_\theta(x)} \sum_{k=1}^{p} P_\theta(x, Y = k) \nabla_\theta \log P_\theta(x, Y = k) \\ &= \sum_{k=1}^{p} P_\theta(Y = k|x)[\nabla_\theta \log \pi_k + \nabla_\theta \log P_\theta(x|Y = k)].\end{aligned}$$

In particular (exercise)

$$\frac{\partial \log P_\theta(x)}{\partial \alpha_k} = P_\theta(Y = k|x) - \pi_k.$$

Relation to classification with generative models (3/3)

The first p elements in the Fisher score are given by class posteriors minus a constant

$$\varphi_\theta(x) = [P_\theta(Y = 1|x) - \pi_1, \ldots, P_\theta(Y = p|x) - \pi_p, ...].$$

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 π_k;

Then,

$$\hat{y}(x) = \operatorname*{argmax}_{k=1,\ldots,p} \varphi_\theta(x)^\top \mathbf{w}_k + b_k$$
$$\hat{y}(x) = \operatorname*{argmax}_{k=1,\ldots,p} P_\theta(Y = k|x).$$

Bayes' rule is implemented via this simple classifier using Fisher kernel.

Outline

5 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 Θ.
- Form the kernel (Seeger, 2002):

$$K\left(\mathbf{x},\mathbf{x}'\right) = \int_{\theta\in\Theta} P_\theta(\mathbf{x})P_\theta(\mathbf{x}')w(d\theta)\ .$$

- No explicit computation of a finite-dimensional feature vector.
- $K\left(\mathbf{x},\mathbf{x}'\right) =< \varphi\left(\mathbf{x}\right), \varphi\left(\mathbf{x}'\right) >_{L_2(w)}$ with

$$\varphi\left(\mathbf{x}\right) = \left(P_\theta\left(\mathbf{x}\right)\right)_{\theta\in\Theta}\ .$$

Example: coin toss

- Let $P_\theta(X=1)=\theta$ and $P_\theta(X=0)=1-\theta$ a model for random coin toss, with $\theta \in [0,1]$.
- Let $d\theta$ be the Lebesgue measure on $[0,1]$
- The mutual information kernel between $\mathbf{x}=001$ and $\mathbf{x}'=1010$ is:

$$\begin{cases} P_\theta\left(\mathbf{x}\right) & =\theta\left(1-\theta\right)^2 \ , \\ P_\theta\left(\mathbf{x}'\right) & =\theta^2\left(1-\theta\right)^2 \ , \end{cases}$$

$$K\left(\mathbf{x},\mathbf{x}'\right)=\int_0^1 \theta^3\left(1-\theta\right)^4 d\theta=\frac{3!4!}{8!}=\frac{1}{280} \, .$$

Outline

5 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 $\mathbf{x} \in \mathcal{X}$, let a latent variable $\mathbf{y} \in \mathcal{Y}$ be associated probabilistically through a conditional probability $P_{\mathbf{x}}(d\mathbf{y})$.
- Let $K_{\mathcal{Z}}$ be a kernel for the complete data $\mathbf{z} = (\mathbf{x}, \mathbf{y})$
- Then, the following kernel is a valid kernel on $\mathcal{X}$, called a marginalized kernel (Tsuda et al., 2002):

$$\begin{aligned} K_{\mathcal{X}}\left(\mathbf{x}, \mathbf{x}'\right) &:= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) \\ &= \int \int K_{\mathcal{Z}}\left((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')\right) P_{\mathbf{x}}(d\mathbf{y}) P_{\mathbf{x}'}\left(d\mathbf{y}'\right) \ . \end{aligned}$$

Marginalized kernels: proof of positive definiteness

- $K_{\mathcal{Z}}$ is p.d. on $\mathcal{Z}$. Therefore, there exists a Hilbert space $\mathcal{H}$ and $\Phi_{\mathcal{Z}} : \mathcal{Z} \to \mathcal{H}$ such that:

$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = \left\langle \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right) \right\rangle_{\mathcal{H}} \ .$$

- Marginalizing therefore gives:

$$\begin{aligned} K_{\mathcal{X}}\left(\mathbf{x}, \mathbf{x}'\right) &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) \\ &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} \left\langle \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right) \right\rangle_{\mathcal{H}} \\ &= \left\langle E_{P_{\mathbf{x}}(d\mathbf{y})} \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), E_{P_{\mathbf{x}'}(d\mathbf{y}')} \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right) \right\rangle_{\mathcal{H}} \ , \end{aligned}$$

therefore $K_{\mathcal{X}}$ is p.d. on $\mathcal{X}$. $\square$

Marginalized kernels: proof of positive definiteness

- $K_{\mathcal{Z}}$ is p.d. on $\mathcal{Z}$. Therefore, there exists a Hilbert space $\mathcal{H}$ and $\Phi_{\mathcal{Z}} : \mathcal{Z} \to \mathcal{H}$ such that:

$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = \left\langle \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right) \right\rangle_{\mathcal{H}} \ .$$

- Marginalizing therefore gives:

$$\begin{aligned} K_{\mathcal{X}}\left(\mathbf{x}, \mathbf{x}'\right) &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) \\ &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} \left\langle \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right) \right\rangle_{\mathcal{H}} \\ &= \left\langle E_{P_{\mathbf{x}}(d\mathbf{y})} \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), E_{P_{\mathbf{x}'}(d\mathbf{y}')} \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right) \right\rangle_{\mathcal{H}} \ , \end{aligned}$$

  therefore $K_{\mathcal{X}}$ 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

4. Kernel Methods: Unsupervised 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

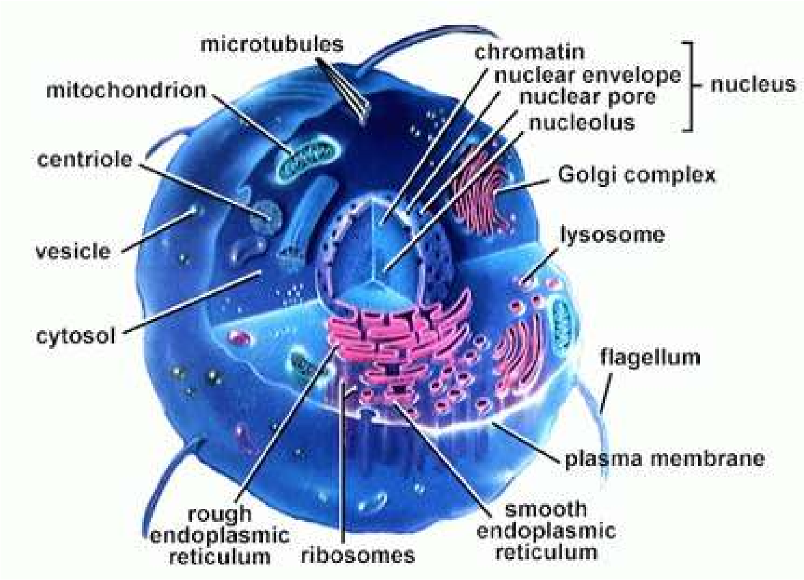
6. Open Problems and Research Topics

Outline

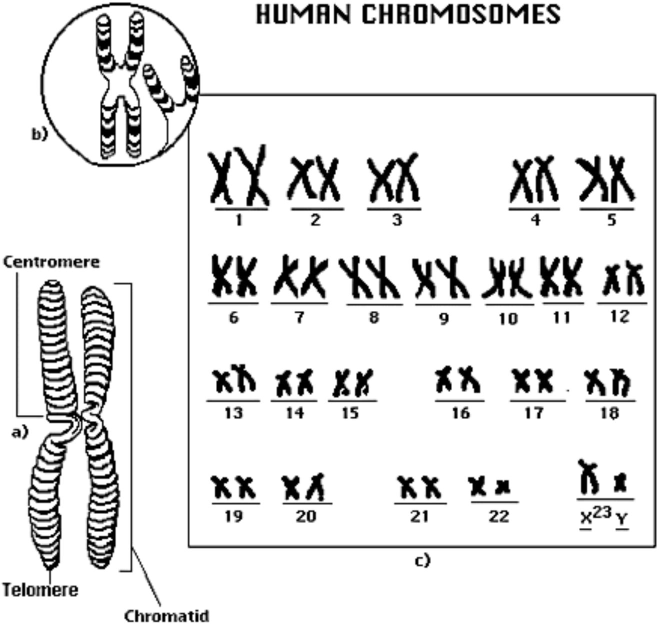
Short history of genomics

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

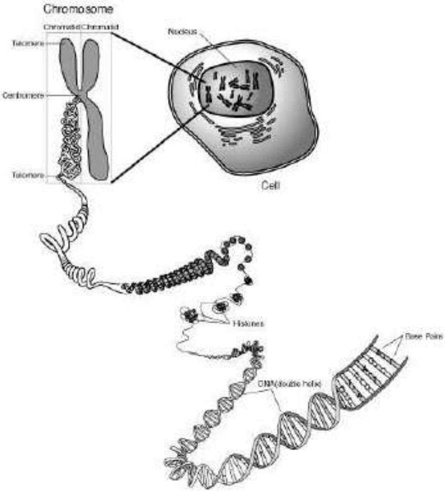
A cell


microtubules
mitochondrion
centriole
vesicle
cytosol
chromatin
nuclear envelope
nuclear pore
nucleolus
nucleus
Golgi complex
lysosome
flagellum
plasma membrane
rough endoplasmic reticulum
ribosomes
smooth endoplasmic reticulum

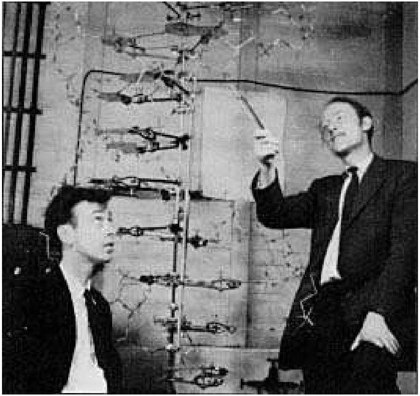
Chromosomes



Chromosomes and DNA



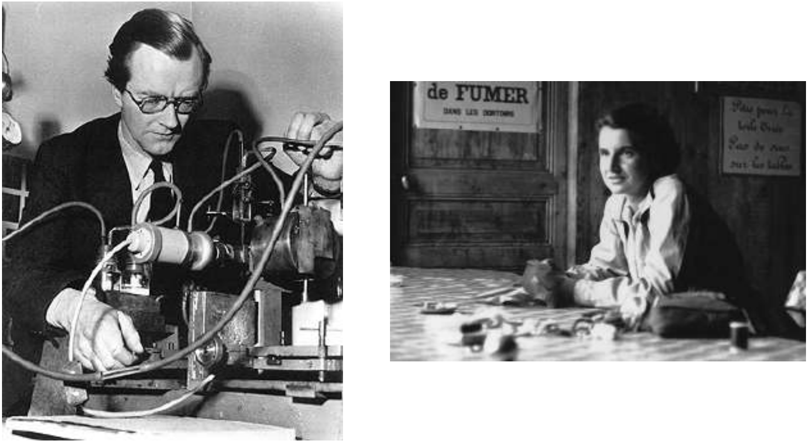
Structure of DNA



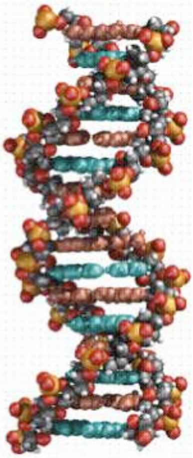
"We wish to suggest a structure for the salt of desoxyribose nucleic acid (D.N.A.). This structure have novel features which are of considerable biological interest" (Watson and Crick, 1953).

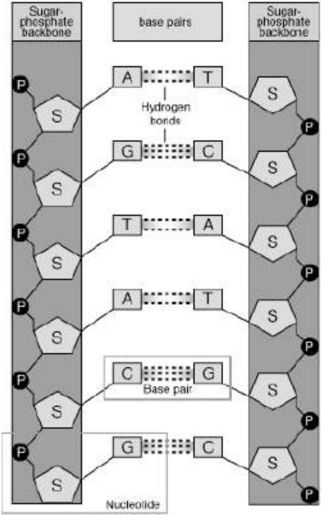
James Watson, Francis Crick, and Maurice Wilkins received the Nobel prize for this discovery in 1962. Key to this discovery were the X-ray crystallography images obtained by Rosalind Franklin.

Structure of DNA

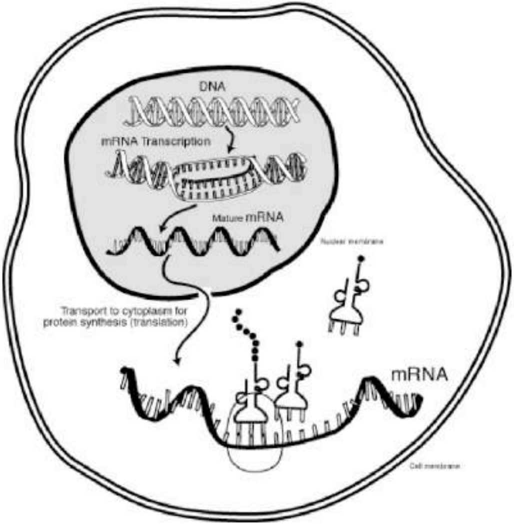


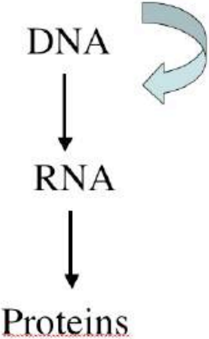
The double helix





Central dogma



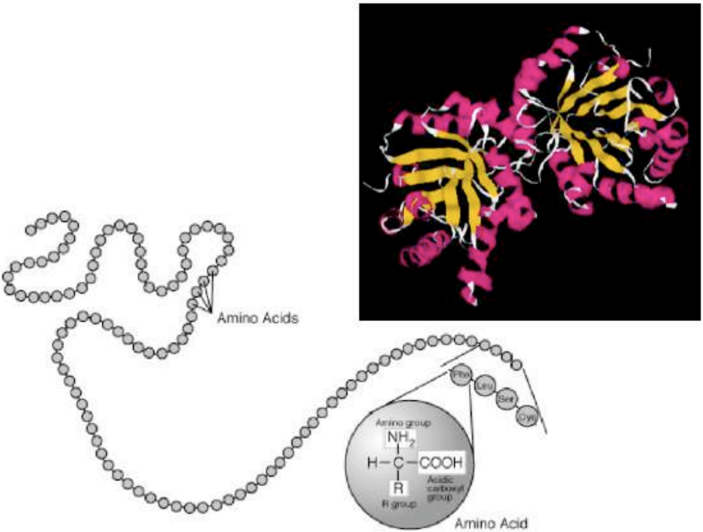


DNA

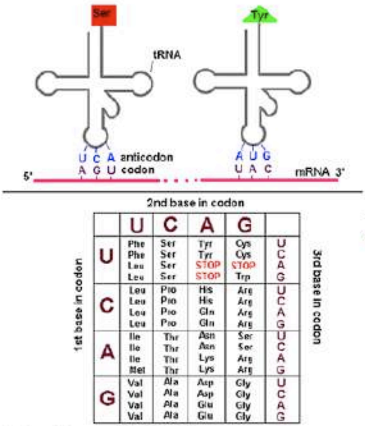
RNA

Proteins

Proteins


Amino Acids
Phe
Leu
Ser
Cys
Amino group
NH2
H—C—COOH
R
Acidic carboxyl group
R group
Amino Acid

Genetic code



2nd base in codon

	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

The Genetic Code

DNA = 4 letters (ATCG)

↓

RNA = 4 letters (AUCG)

↓

Protein = 20 letters (amino acids)

1 amino acid

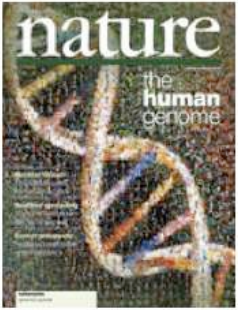
=

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

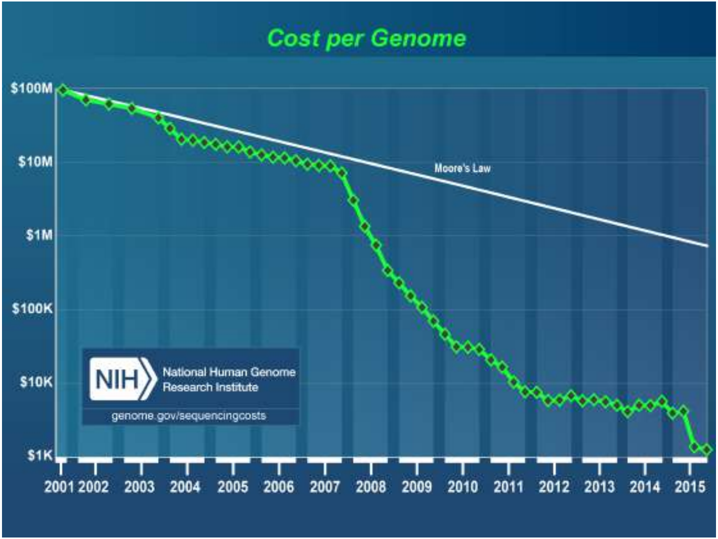
2003: End of genomics era



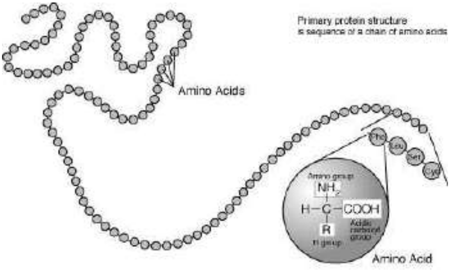
Findings

- 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



Protein sequence



A : Alanine
F : Phenylalanine
E : Glutamic acid
T : Threonine
H : Histidine
I : Isoleucine
D : Aspartic acid

V : Valine
P : Proline
K : Lysine
C : Cysteine
Y : Tyrosine
S : Serine
G : Glycine

L : Leucine
M : Methionine
R : Arginine
N : Asparagine
W : Tryptophane
Q : Glutamine

Challenges with protein sequences

- A protein sequences can be seen as a variable-length sequence over the 20-letter alphabet of amino-acids, e.g., insuline: `FVNQHLCGSHLVEALYLVCGERGFFYTPKA`
- These sequences are produced at a fast rate (result of the sequencing programs)
- Need for algorithms to compare, classify, analyze these sequences
- Applications: classification into functional or structural classes, prediction of cellular localization and interactions, ...

Example: supervised sequence classification

Data (training)

- Secreted proteins:
  `MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNIEA...`
  `MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...`
  `MALHTVLIMLSLLPMLEAQNPEHANITIGEPITNETLGWL...`
  ...
- Non-secreted proteins:
  `MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVNLGVG...`
  `MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...`
  `MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..`
  ...

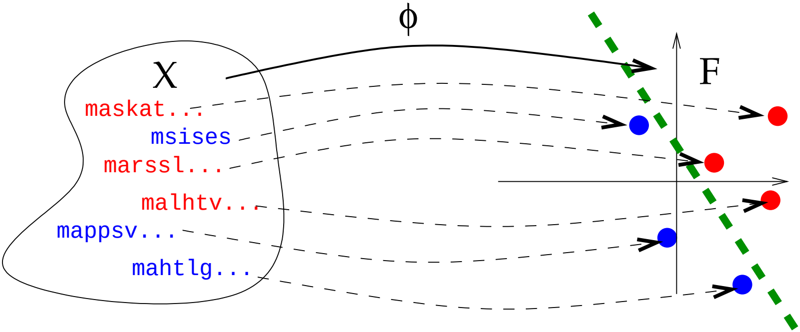
Goal

- Build a classifier to predict whether new proteins are secreted or not.

Supervised classification with vector embedding

The idea

- Map each string $\mathbf{x} \in \mathcal{X}$ to a vector $\Phi(\mathbf{x}) \in \mathcal{F}$.
- Train a classifier for vectors on the images $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)$ of the training set (nearest neighbor, linear perceptron, logistic regression, support vector machine...)



Kernels for protein sequences

- Kernel methods have been widely investigated since Jaakkola et al.'s seminal paper (1998).
- What is a good kernel?
  - it should be mathematically valid (symmetric, p.d. or c.p.d.)
  - fast to compute
  - adapted to the problem (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

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

Vector embedding for strings

The idea

Represent each sequence **x** by a fixed-length numerical vector $\Phi(\mathbf{x}) \in \mathbb{R}^n$. How to perform this embedding?

Vector embedding for strings

The idea

Represent each sequence $\mathbf{x}$ by a fixed-length numerical vector $\Phi(\mathbf{x}) \in \mathbb{R}^n$. How to perform this embedding?

Physico-chemical kernel

Extract relevant features, such as:

- length of the sequence
- time series analysis of numerical physico-chemical properties of amino-acids along the sequence (e.g., polarity, hydrophobicity), using for example:
  - Fourier transforms (Wang et al., 2004)
  - Autocorrelation functions (Zhang et al., 2003)

$$r_j = \frac{1}{n-j} \sum_{i=1}^{n-j} h_i h_{i+j}$$

Substring indexation

The approach

Alternatively, index the feature space by fixed-length strings, i.e.,

$$\Phi(\mathbf{x}) = (\Phi_u(\mathbf{x}))_{u \in \mathcal{A}^k}$$

where $\Phi_u(\mathbf{x})$ can be:

- the number of occurrences of u in $\mathbf{x}$ (without gaps) : spectrum kernel (Leslie et al., 2002)
- the number of occurrences of u in $\mathbf{x}$ up to m mismatches (without gaps) : mismatch kernel (Leslie et al., 2004)
- the number of occurrences of u in $\mathbf{x}$ allowing gaps, with a weight decaying exponentially with the number of gaps : substring kernel (Lohdi et al., 2002)

Example: Spectrum kernel (1/4)

Kernel definition

- The 3-spectrum of

$$\mathbf{x} = \texttt{CGGSLIAMMWFGV}$$

  is:

$$(\texttt{CGG,GGS,GSL,SLI,LIA,IAM,AMM,MMW,MWF,WFG,FGV})\ .$$

- Let $\Phi_u(\mathbf{x})$ denote the number of occurrences of u in $\mathbf{x}$. The k-spectrum kernel is:

$$K\left(\mathbf{x},\mathbf{x}'\right) := \sum_{u\in\mathcal{A}^k} \Phi_u\left(\mathbf{x}\right)\Phi_u\left(\mathbf{x}'\right)\ .$$

Example: Spectrum kernel (2/4)

Implementation

- The computation of the kernel is formally a sum over $|\mathcal{A}|^k$ terms, but at most $|\mathbf{x}| - k + 1$ terms are non-zero in $\Phi(\mathbf{x}) \implies$ Computation in $O(|\mathbf{x}| + |\mathbf{x}'|)$ with pre-indexation of the strings.
- Fast classification of a sequence $\mathbf{x}$ in $O(|\mathbf{x}|)$:

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_u w_u \Phi_u(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|-k+1} w_{x_i \dots x_{i+k-1}}.$$

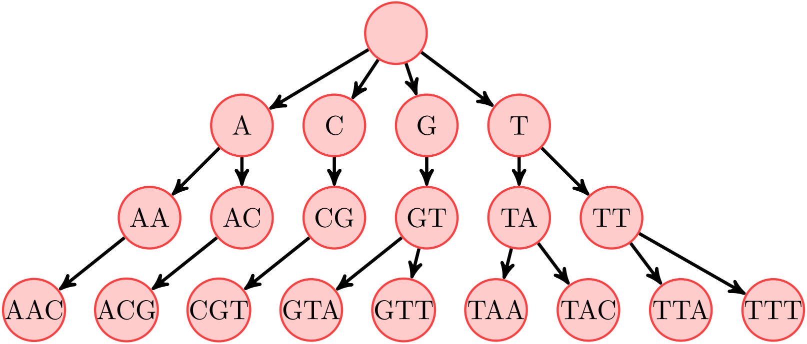
Remarks

- Work with any string (natural language, time series...)
- Fast and scalable, a good default method for string classification.
- Variants allow matching of k-mers up to m mismatches.

Example: Spectrum kernel (3/4)

If pre-indexation is not possible: retrieval tree (trie)

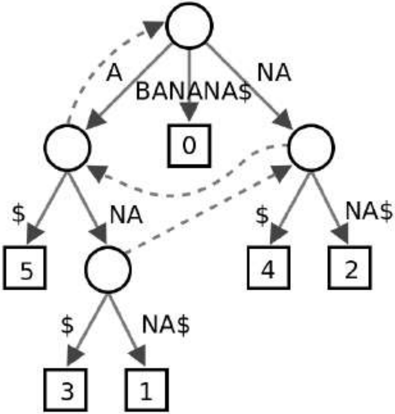
Consider the sequence `ACGTTTAACGTAC`.



The complexity for computing $K(\mathbf{x}, \mathbf{x}')$ becomes $O(k(|\mathbf{x}| + |\mathbf{x}'|))$.

Example: Spectrum kernel (4/4)

If pre-indexation is not possible: use a suffix tree



The complexity for computing $K(\mathbf{x}, \mathbf{x}')$ becomes $O(|\mathbf{x}| + |\mathbf{x}'|)$, but with a larger constant than with pre-indexation.

Example 2: Substring kernel (1/12)

Definition

- For $1 \leq k \leq n \in \mathbb{N}$, we denote by $\mathcal{I}(k, n)$ the set of sequences of indices $\mathbf{i} = (i_1, \ldots, i_k)$, with $1 \leq i_1 < i_2 < \ldots < i_k \leq n$.
- For a string $\mathbf{x} = x_1 \ldots x_n \in \mathcal{X}$ of length n, for a sequence of indices $\mathbf{i} \in \mathcal{I}(k, n)$, we define a substring as:

$$\mathbf{x}(\mathbf{i}) := x_{i_1} x_{i_2} \ldots x_{i_k} .$$

- The length of the substring is:

$$l(\mathbf{i}) = i_k - i_1 + 1.$$

Example 2: Substring kernel (2/12)

Example

ABRACADABRA

- $\mathbf{i} = (3, 4, 7, 8, 10)$
- $\mathbf{x}(\mathbf{i}) =$ RADAR
- $l(\mathbf{i}) = 10 - 3 + 1 = 8$

Example 2: Substring kernel (3/12)

The kernel

- Let $k \in \mathbb{N}$ and $\lambda \in \mathbb{R}^+$ fixed. For all $\mathbf{u} \in \mathcal{A}^k$, let $\Phi_{\mathbf{u}} : \mathcal{X} \to \mathbb{R}$ be defined by:

$$\forall \mathbf{x} \in \mathcal{X}, \quad \Phi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i} \in \mathcal{I}(k, |\mathbf{x}|): \quad \mathbf{x}(\mathbf{i}) = \mathbf{u}} \lambda^{l(\mathbf{i})} .$$

- The substring kernel is the p.d. kernel defined by:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K_{k,\lambda}(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{u} \in \mathcal{A}^k} \Phi_{\mathbf{u}}(\mathbf{x}) \Phi_{\mathbf{u}}(\mathbf{x}') .$$

Example 2: Substring kernel (4/12)

Example

u	ca	ct	at	ba	bt	cr	ar	br
$\Phi_u(\text{cat})$	λ^2	λ^3	λ^2	0	0	0	0	0
$\Phi_u(\text{car})$	λ^2	0	0	0	0	λ^3	λ^2	0
$\Phi_u(\text{bat})$	0	0	λ^2	λ^2	λ^3	0	0	0
$\Phi_u(\text{bar})$	0	0	0	λ^2	0	0	λ^2	λ^3

$$\begin{cases} K\,(\text{cat,cat}) = K\,(\text{car,car}) = 2\lambda^4 + \lambda^6 \\ K\,(\text{cat,car}) = \lambda^4 \\ K\,(\text{cat,bar}) = 0 \end{cases}$$

Example 2: Substring kernel (5/12)

Kernel computation

- We need to compute, for any pair $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, the kernel:

$$
\begin{aligned}
K_{k,\lambda}\left(\mathbf{x}, \mathbf{x}'\right) &= \sum_{\mathbf{u} \in \mathcal{A}^k} \Phi_{\mathbf{u}}\left(\mathbf{x}\right) \Phi_{\mathbf{u}}\left(\mathbf{x}'\right) \\
&= \sum_{\mathbf{u} \in \mathcal{A}^k} \sum_{\mathbf{i}:\mathbf{x}(\mathbf{i})=\mathbf{u}} \sum_{\mathbf{i}':\mathbf{x}'(\mathbf{i}')=\mathbf{u}} \lambda^{l(\mathbf{i})+l(\mathbf{i}')} .
\end{aligned}
$$

- Enumerating the substrings is too slow (of order $|\mathbf{x}|^k$).

Example 2: Substring kernel (6/12)

Kernel computation (cont.)

- For $\mathbf{u} \in \mathcal{A}^k$ remember that:

$$\Phi_{\mathbf{u}}\left(\mathbf{x}\right) = \sum_{\mathbf{i}:\mathbf{x}(\mathbf{i})=\mathbf{u}} \lambda^{i_k - i_1 + 1} .$$

- Let now:

$$\Psi_{\mathbf{u}}\left(\mathbf{x}\right) = \sum_{\mathbf{i}:\mathbf{x}(\mathbf{i})=\mathbf{u}} \lambda^{|\mathbf{x}| - i_1 + 1} .$$

Example 2: Substring kernel (7/12)

Kernel computation (cont.)

Let us note $\mathbf{x}_{[1,j]} = x_1 \dots x_j$. A simple rewriting shows that, if we note $a \in \mathcal{A}$ the last letter of $\mathbf{u}$ ($\mathbf{u} = \mathbf{v}a$):

$$\Phi_{\mathbf{v}a}(\mathbf{x}) = \sum_{j \in [1, |\mathbf{x}|] : x_j = a} \Psi_{\mathbf{v}}\left(\mathbf{x}_{[1,j-1]}\right) \lambda \,,$$

and

$$\Psi_{\mathbf{v}a}(\mathbf{x}) = \sum_{j \in [1, |\mathbf{x}|] : x_j = a} \Psi_{\mathbf{v}}\left(\mathbf{x}_{[1,j-1]}\right) \lambda^{|\mathbf{x}|-j+1} \,.$$

Example 2: Substring kernel (8/12)

Kernel computation (cont.)

Moreover we observe that if the string is of the form $\mathbf{x}a$ (i.e., the last letter is $a \in \mathcal{A}$), then:

- If the last letter of $\mathbf{u}$ is not a:

$$\begin{cases} \Phi_{\mathbf{u}}(\mathbf{x}a) & = \Phi_{\mathbf{u}}(\mathbf{x}) \; , \\ \Psi_{\mathbf{u}}(\mathbf{x}a) & = \lambda \Psi_{\mathbf{u}}(\mathbf{x}) \; . \end{cases}$$

- If the last letter of $\mathbf{u}$ is a (i.e., $\mathbf{u} = \mathbf{v}a$ with $\mathbf{v} \in \mathcal{A}^{k-1}$):

$$\begin{cases} \Phi_{\mathbf{v}a}(\mathbf{x}a) & = \Phi_{\mathbf{v}a}(\mathbf{x}) + \lambda \Psi_{\mathbf{v}}(\mathbf{x}) \; , \\ \Psi_{\mathbf{v}a}(\mathbf{x}a) & = \lambda \Psi_{\mathbf{v}a}(\mathbf{x}) + \lambda \Psi_{\mathbf{v}}(\mathbf{x}) \; . \end{cases}$$

Example 2: Substring kernel (9/12)

Kernel computation (cont.)

Let us now show how the function:

$$B_k\left(\mathbf{x},\mathbf{x}'\right) := \sum_{\mathbf{u}\in\mathcal{A}^k} \Psi_{\mathbf{u}}\left(\mathbf{x}\right)\Psi_{\mathbf{u}}\left(\mathbf{x}'\right)$$

and the kernel:

$$K_k\left(\mathbf{x},\mathbf{x}'\right) := \sum_{\mathbf{u}\in\mathcal{A}^k} \Phi_{\mathbf{u}}\left(\mathbf{x}\right)\Phi_{\mathbf{u}}\left(\mathbf{x}'\right)$$

can be computed recursively. We note that:

$$\begin{cases} B_0\left(\mathbf{x},\mathbf{x}'\right) = K_0\left(\mathbf{x},\mathbf{x}'\right) = 1 & \text{for all } \mathbf{x},\mathbf{x}' \\ B_k\left(\mathbf{x},\mathbf{x}'\right) = K_k\left(\mathbf{x},\mathbf{x}'\right) = 0 & \text{if } \min\left(|\,\mathbf{x}\,|\,,|\,\mathbf{x}'\,|\right) < k \end{cases}$$

Example 2: Substring kernel (10/12)

Recursive computation of B_k

$$
\begin{aligned}
& B_k\left(\mathbf{x}a, \mathbf{x}'\right) \\
& = \sum_{\mathbf{u} \in \mathcal{A}^k} \Psi_{\mathbf{u}}\left(\mathbf{x}a\right) \Psi_{\mathbf{u}}\left(\mathbf{x}'\right) \\
& = \lambda \sum_{\mathbf{u} \in \mathcal{A}^k} \Psi_{\mathbf{u}}\left(\mathbf{x}\right) \Psi_{\mathbf{u}}\left(\mathbf{x}'\right) + \lambda \sum_{\mathbf{v} \in \mathcal{A}^{k-1}} \Psi_{\mathbf{v}}\left(\mathbf{x}\right) \Psi_{\mathbf{v}a}\left(\mathbf{x}'\right) \\
& = \lambda B_k\left(\mathbf{x}, \mathbf{x}'\right) + \\
& \quad \lambda \sum_{\mathbf{v} \in \mathcal{A}^{k-1}} \Psi_{\mathbf{v}}\left(\mathbf{x}\right) \left(\sum_{j \in [1, |\mathbf{x}'|] : x'_j = a} \Psi_{\mathbf{v}}\left(\mathbf{x}'_{[1,j-1]}\right) \lambda^{|\mathbf{x}'| - j + 1} \right) \\
& = \lambda B_k\left(\mathbf{x}, \mathbf{x}'\right) + \sum_{j \in [1, |\mathbf{x}'|] : x'_j = a} B_{k-1}\left(\mathbf{x}, \mathbf{x}'_{[1,j-1]}\right) \lambda^{|\mathbf{x}'| - j + 2}
\end{aligned}
$$

Example 2: Substring kernel (11/12)

Recursive computation of B_k

$$
\begin{aligned}
& B_k\left(\mathbf{x}a, \mathbf{x}'b\right) \\
& = \lambda B_k\left(\mathbf{x}, \mathbf{x}'b\right) + \lambda \sum_{j \in [1, |\mathbf{x}'|] : x'_j = a} B_{k-1}\left(\mathbf{x}, \mathbf{x}'_{[1,j-1]}\right) \lambda^{|\mathbf{x}'| - j + 2} \\
& \qquad + \delta_{a=b} B_{k-1}(\mathbf{x}, \mathbf{x}') \lambda^2 \\
& = \lambda B_k\left(\mathbf{x}, \mathbf{x}'b\right) + \lambda(B_k(\mathbf{x}a, \mathbf{x}') - \lambda B_k(\mathbf{x}, \mathbf{x}')) + \delta_{a=b} B_{k-1}(\mathbf{x}, \mathbf{x}') \lambda^2 \\
& = \lambda B_k\left(\mathbf{x}, \mathbf{x}'b\right) + \lambda B_k(\mathbf{x}a, \mathbf{x}') - \lambda^2 B_k(\mathbf{x}, \mathbf{x}') + \delta_{a=b} B_{k-1}(\mathbf{x}, \mathbf{x}') \lambda^2 .
\end{aligned}
$$

The dynamic programming table can be filled in $O(k|\mathbf{x}||\mathbf{x}'|)$ operations.

Example 2: Substring kernel (12/12)

Recursive computation of K_k

$$
\begin{aligned}
&K_k\left(\mathbf{x}a, \mathbf{x}'\right) \\
&= \sum_{\mathbf{u}\in\mathcal{A}^k} \Phi_{\mathbf{u}}\left(\mathbf{x}a\right)\Phi_{\mathbf{u}}\left(\mathbf{x}'\right) \\
&= \sum_{\mathbf{u}\in\mathcal{A}^k} \Phi_{\mathbf{u}}\left(\mathbf{x}\right)\Phi_{\mathbf{u}}\left(\mathbf{x}'\right) + \lambda \sum_{\mathbf{v}\in\mathcal{A}^{k-1}} \Psi_{\mathbf{v}}\left(\mathbf{x}\right)\Phi_{\mathbf{v}a}\left(\mathbf{x}'\right) \\
&= K_k\left(\mathbf{x}, \mathbf{x}'\right) + \\
&\quad \lambda \sum_{\mathbf{v}\in\mathcal{A}^{k-1}} \Psi_{\mathbf{v}}\left(\mathbf{x}\right)\left(\sum_{j\in[1,|\mathbf{x}'|]:x'_j=a} \Psi_{\mathbf{v}}\left(\mathbf{x}'_{[1,j-1]}\right)\lambda\right) \\
&= K_k\left(\mathbf{x}, \mathbf{x}'\right) + \lambda^2 \sum_{j\in[1,|\mathbf{x}'|]:x'_j=a} B_{k-1}\left(\mathbf{x}, \mathbf{x}'_{[1,j-1]}\right)
\end{aligned}
$$

Summary: Substring indexation

- Implementation in $O(|\mathbf{x}| + |\mathbf{x}'|)$ in memory and time for the spectrum and mismatch kernels (with suffix trees)
- Implementation in $O(k(|\mathbf{x}| + |\mathbf{x}'|))$ in memory and time for the spectrum and mismatch kernels (with tries)
- Implementation in $O(k|\mathbf{x}| \times |\mathbf{x}'|)$ in memory and time for the substring kernels
- The feature space has high dimension ($|\mathcal{A}|^k$), so learning requires regularized methods (such as SVM)

Dictionary-based indexation

The approach

- Chose a dictionary of sequences $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$
- Chose a measure of similarity $s(\mathbf{x}, \mathbf{x}')$
- Define the mapping $\Phi_{\mathcal{D}}(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_i))_{\mathbf{x}_i \in \mathcal{D}}$

Dictionary-based indexation

The approach

- Chose a dictionary of sequences $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$
- Chose a measure of similarity $s(\mathbf{x}, \mathbf{x}')$
- Define the mapping $\Phi_{\mathcal{D}}(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_i))_{\mathbf{x}_i \in \mathcal{D}}$

Examples

This includes:

- Motif kernels (Logan et al., 2001): the dictionary is a library of motifs, the similarity function is a matching function
- Pairwise kernel (Liao & Noble, 2003): the dictionary is the training set, the similarity is a classical measure of similarity between sequences.

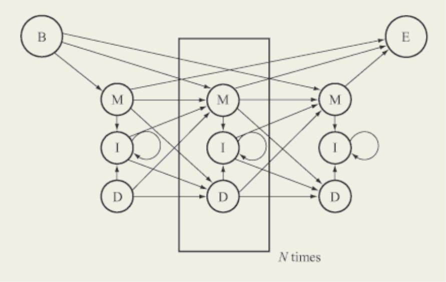
Outline

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

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

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

Context-tree model

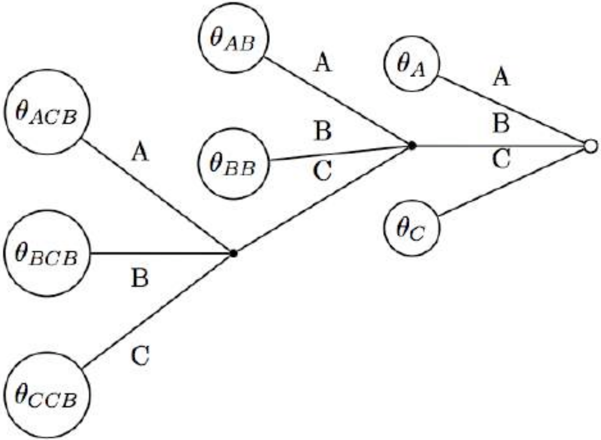
Definition

A context-tree model is a variable-memory Markov chain:

$$P_{\mathcal{D},\theta}(\mathbf{x}) = P_{\mathcal{D},\theta}\left(x_1 \ldots x_D\right) \prod_{i=D+1}^{n} P_{\mathcal{D},\theta}\left(x_i \,|\, x_{i-D} \ldots x_{i-1}\right)$$

- $\mathcal{D}$ is a suffix tree
- $\theta \in \Sigma^{\mathcal{D}}$ is a set of conditional probabilities (multinomials)

Context-tree model: example



$$P(AABACBACC) = P(AAB)\theta_{AB}(A)\theta_A(C)\theta_C(B)\theta_{ACB}(A)\theta_A(C)\theta_C(A)\ .$$

The context-tree kernel

Theorem (Cuturi et al., 2005)

- *For particular choices of priors, the context-tree kernel:*

$$K\left(\mathbf{x},\mathbf{x}'\right)=\sum_{\mathcal{D}}\int_{\theta\in\Sigma^{\mathcal{D}}}P_{\mathcal{D},\theta}(\mathbf{x})P_{\mathcal{D},\theta}(\mathbf{x}')w(d\theta|\mathcal{D})\pi(\mathcal{D})$$

  can be computed in $O(|\mathbf{x}|+|\mathbf{x}'|)$ with a variant of the Context-Tree Weighting algorithm.
- *This is a valid mutual information kernel.*
- *The similarity is related to information-theoretical measure of mutual information between strings.*

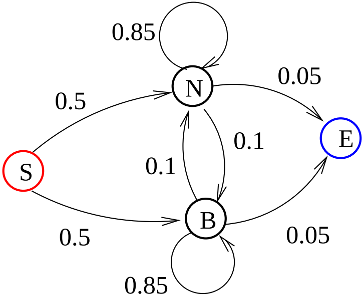
Marginalized kernels

Recall: Definition

- For any observed data $\mathbf{x} \in \mathcal{X}$, let a latent variable $\mathbf{y} \in \mathcal{Y}$ be associated probabilistically through a conditional probability $P_{\mathbf{x}}(d\mathbf{y})$.
- Let $K_{\mathcal{Z}}$ be a kernel for the complete data $\mathbf{z} = (\mathbf{x}, \mathbf{y})$
- Then the following kernel is a valid kernel on $\mathcal{X}$, called a marginalized kernel (Tsuda et al., 2002):

$$\begin{aligned} K_{\mathcal{X}}\left(\mathbf{x}, \mathbf{x}'\right) &:= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) \\ &= \int\int K_{\mathcal{Z}}\left((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')\right) P_{\mathbf{x}}(d\mathbf{y}) P_{\mathbf{x}'}\left(d\mathbf{y}'\right) . \end{aligned}$$

Example: HMM for normal/biased coin toss



- Normal (N) and biased (B) coins (not observed)
- Observed output are 0/1 with probabilities:

$$\begin{cases} \pi(0|N) = 1 - \pi(1|N) = 0.5, \\ \pi(0|B) = 1 - \pi(1|B) = 0.2. \end{cases}$$

- Example of realization (complete data):

NNNNNBBBBBBBBBNNNNNNNNNNNBBBBBB
1001011101111010010111001111011

1-spectrum kernel on complete data

- If both $\mathbf{x} \in \mathcal{A}^*$ and $\mathbf{y} \in \mathcal{S}^*$ were observed, we might rather use the 1-spectrum kernel on the complete data $\mathbf{z} = (\mathbf{x}, \mathbf{y})$:

$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = \sum_{(a,s) \in \mathcal{A} \times \mathcal{S}} n_{a,s}\left(\mathbf{z}\right) n_{a,s}\left(\mathbf{z}'\right),$$

  where $n_{a,s}\left(\mathbf{x}, \mathbf{y}\right)$ for $a = 0, 1$ and $s = N, B$ is the number of occurrences of s in $\mathbf{y}$ which emit a in $\mathbf{x}$.

- Example:

$$\mathbf{z} = 1001011101111010010111001111011,$$
$$\mathbf{z}' = 0011010110011111011010111101100101,$$

$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = n_1\left(\mathbf{z}\right) n_1\left(\mathbf{z}'\right) + n_1\left(\mathbf{z}\right) n_1\left(\mathbf{z}'\right) + n_0\left(\mathbf{z}\right) n_0\left(\mathbf{z}'\right) + n_0\left(\mathbf{z}\right) n_0\left(\mathbf{z}'\right)$$
$$= 7 \times 15 + 13 \times 6 + 9 \times 12 + 2 \times 1 = 293.$$

1-spectrum marginalized kernel on observed data

- The marginalized kernel for observed data is:

$$K_{\mathcal{X}}\left(\mathbf{x},\mathbf{x}'\right) = \sum_{\mathbf{y},\mathbf{y}'\in\mathcal{S}^*} K_{\mathcal{Z}}\left(\left(\mathbf{x},\mathbf{y}\right),\left(\mathbf{x}',\mathbf{y}'\right)\right) P\left(\mathbf{y}|\mathbf{x}\right) P\left(\mathbf{y}'|\mathbf{x}'\right)$$

$$= \sum_{(a,s)\in\mathcal{A}\times\mathcal{S}} \Phi_{a,s}\left(\mathbf{x}\right)\Phi_{a,s}\left(\mathbf{x}'\right),$$

with

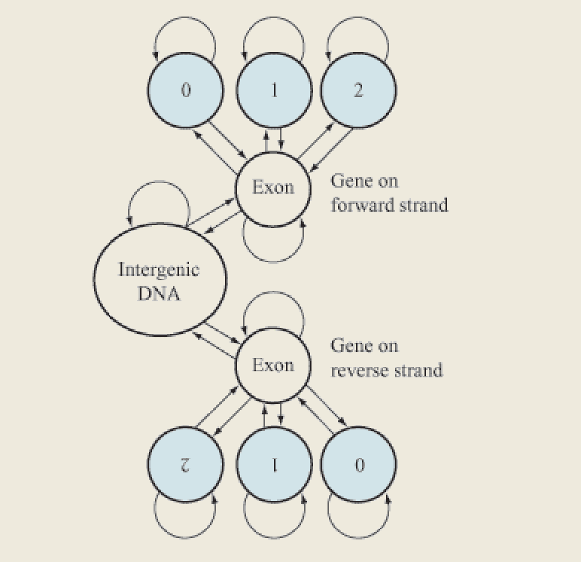
$$\Phi_{a,s}\left(\mathbf{x}\right) = \sum_{\mathbf{y}\in\mathcal{S}^*} P\left(\mathbf{y}|\mathbf{x}\right) n_{a,s}\left(\mathbf{x},\mathbf{y}\right)$$

Computation of the 1-spectrum marginalized kernel

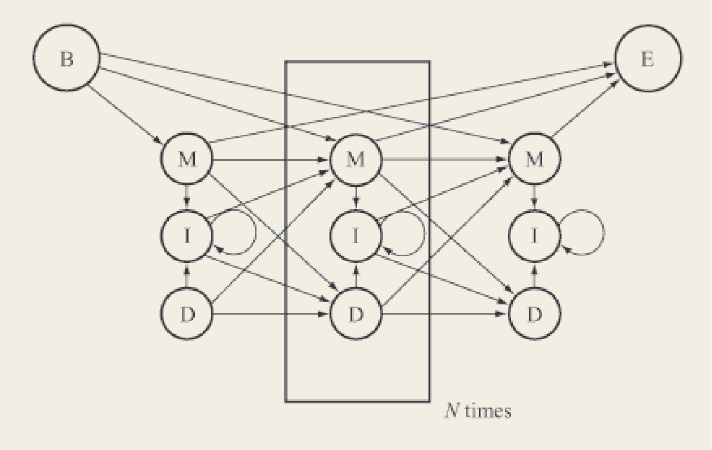
$$\begin{aligned}
\Phi_{a,s}(\mathbf{x}) &= \sum_{\mathbf{y}\in\mathcal{S}^*} P(\mathbf{y}|\mathbf{x})\, n_{a,s}(\mathbf{x},\mathbf{y}) \\
&= \sum_{\mathbf{y}\in\mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) \left\{ \sum_{i=1}^{n} \delta(x_i,a)\,\delta(y_i,s) \right\} \\
&= \sum_{i=1}^{n} \delta(x_i,a) \left\{ \sum_{\mathbf{y}\in\mathcal{S}^*} P(\mathbf{y}|\mathbf{x})\,\delta(y_i,s) \right\} \\
&= \sum_{i=1}^{n} \delta(x_i,a)\, P(y_i = s|\mathbf{x})\,.
\end{aligned}$$

and $P(y_i = s|\mathbf{x})$ can be computed efficiently by forward-backward algorithm!

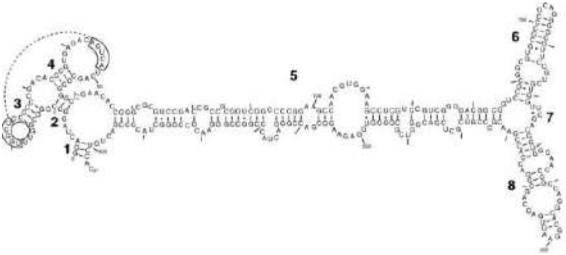
HMM example (DNA)



HMM example (protein)



SCFG for RNA sequences



SFCG rules

- $S \rightarrow SS$
- $S \rightarrow aSa$
- $S \rightarrow aS$
- $S \rightarrow a$

Marginalized kernel (Kin et al., 2002)

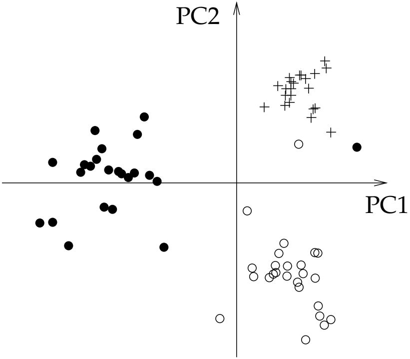
- Feature: number of occurrences of each (base,state) combination
- Marginalization using classical inside/outside algorithm

Marginalized kernels in practice

Examples

- Spectrum kernel on the hidden states of a HMM for protein sequences (Tsuda et al., 2002)
- Kernels for RNA sequences based on SCFG (Kin et al., 2002)
- Kernels for graphs based on random walks on graphs (Kashima et al., 2004)
- Kernels for multiple alignments based on phylogenetic models (Vert et al., 2006)

Marginalized kernels: example



A set of 74 human tRNA sequences is analyzed using a kernel for sequences (the second-order marginalized kernel based on SCFG). This set of tRNAs contains three classes, called Ala-AGC (*white circles*), Asn-GTT (*black circles*) and Cys-GCA (*plus symbols*) (from Tsuda et al., 2002).

Outline

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

Sequence alignment

Motivation

How to compare 2 sequences?

$$\mathbf{x}_1 = \mathtt{CGGSLIAMMWFGV}$$
$$\mathbf{x}_2 = \mathtt{CLIVMMNRLMWFGV}$$

Find a good alignment:

```
CGGSLIAMM------WFGV
 |...|||||....||||
C-----LIVMMNRLMWFGV
```

Alignment score

In order to quantify the relevance of an alignment π, define:

- a substitution matrix $S \in \mathbb{R}^{\mathcal{A} \times \mathcal{A}}$
- a gap penalty function $g : \mathbb{N} \to \mathbb{R}$

Any alignment is then scored as follows

```
CGGSLIAMM------WFGV
 |...|||||....||||
C----LIVMMNRLMWFGV
```

$$\begin{aligned} s_{S,g}(\pi) = {} & S(C,C) + S(L,L) + S(I,I) + S(A,V) + 2S(M,M) \\ & + S(W,W) + S(F,F) + S(G,G) + S(V,V) - g(3) - g(4) \end{aligned}$$

Local alignment kernel

Smith-Waterman score (Smith and Waterman, 1981)

- The widely-used Smith-Waterman local alignment score is defined by:

$$SW_{S,g}(\mathbf{x},\mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x},\mathbf{y})} s_{S,g}(\pi).$$

- It is symmetric, but not positive definite...

Local alignment kernel

Smith-Waterman score (Smith and Waterman, 1981)

- The widely-used Smith-Waterman local alignment score is defined by:

$$SW_{S,g}(\mathbf{x},\mathbf{y}) := \max_{\pi\in\Pi(\mathbf{x},\mathbf{y})} s_{S,g}(\pi).$$

- It is symmetric, but not positive definite...

LA kernel (Saigo et al., 2004)

The local alignment kernel:

$$K_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = \sum_{\pi\in\Pi(\mathbf{x},\mathbf{y})} \exp\left(\beta s_{S,g}\left(\mathbf{x},\mathbf{y},\pi\right)\right),$$

is symmetric positive definite.

LA kernel is p.d.: proof (1/11)

Lemma

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

$$K_1 + K_2,$$
$$K_1 K_2, \text{ and}$$
$$cK_1, \text{ for } c \geq 0,$$

  are also p.d. kernels

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

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

  then K is also a p.d. kernel.

LA kernel is p.d.: proof (2/11)

Proof of lemma

Let A and B be $n \times n$ positive semidefinite matrices. By diagonalization of A:

$$A_{i,j} = \sum_{p=1}^{n} f_p(i) f_p(j)$$

for some vectors $f_1, \ldots, f_n$. Then, for any $\boldsymbol{\alpha} \in \mathbb{R}^n$:

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j A_{i,j} B_{i,j} = \sum_{p=1}^{n} \sum_{i,j=1}^{n} \alpha_i f_p(i) \alpha_j f_p(j) B_{i,j} \geq 0.$$

The matrix $C_{i,j} = A_{i,j} B_{i,j}$ is therefore p.d. Other properties are obvious from definition. $\square$

LA kernel is p.d.: proof (3/11)

Lemma (direct sum and product of kernels)

Let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$. Let K_1 be a p.d. kernel on $\mathcal{X}_1$, and K_2 be a p.d. kernel on $\mathcal{X}_2$. Then the following functions are p.d. kernels on $\mathcal{X}$:

- the direct sum,

$$K\left(\left(\mathbf{x}_1, \mathbf{x}_2\right), \left(\mathbf{y}_1, \mathbf{y}_2\right)\right) = K_1\left(\mathbf{x}_1, \mathbf{y}_1\right) + K_2\left(\mathbf{x}_2, \mathbf{y}_2\right),$$

- The direct product:

$$K\left(\left(\mathbf{x}_1, \mathbf{x}_2\right), \left(\mathbf{y}_1, \mathbf{y}_2\right)\right) = K_1\left(\mathbf{x}_1, \mathbf{y}_1\right) K_2\left(\mathbf{x}_2, \mathbf{y}_2\right).$$

LA kernel is p.d.: proof (4/11)

Proof of lemma

If K_1 is a p.d. kernel, let $\Phi_1 : \mathcal{X}_1 \mapsto \mathcal{H}$ be such that:

$$K_1\left(\mathbf{x}_1, \mathbf{y}_1\right) = \left\langle \Phi_1\left(\mathbf{x}_1\right), \Phi_1\left(\mathbf{y}_1\right)\right\rangle_{\mathcal{H}}.$$

Let $\Phi : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{H}$ be defined by:

$$\Phi\left(\left(\mathbf{x}_1, \mathbf{x}_2\right)\right) = \Phi_1\left(\mathbf{x}_1\right).$$

Then for $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ and $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2) \in \mathcal{X}$, we get

$$\left\langle \Phi\left(\left(\mathbf{x}_1, \mathbf{x}_2\right)\right), \Phi\left(\left(\mathbf{y}_1, \mathbf{y}_2\right)\right)\right\rangle_{\mathcal{H}} = K_1\left(\mathbf{x}_1, \mathbf{x}_2\right),$$

which shows that $K\left(\mathbf{x}, \mathbf{y}\right) := K_1\left(\mathbf{x}_1, \mathbf{y}_1\right)$ is p.d. on $\mathcal{X}_1 \times \mathcal{X}_2$. The lemma follows from the properties of sums and products of p.d. kernels. $\square$

LA kernel is p.d.: proof (5/11)

Lemma: kernel for sets

Let K be a p.d. kernel on $\mathcal{X}$, and let $\mathcal{P}(\mathcal{X})$ be the set of finite subsets of $\mathcal{X}$. Then the function K_P on $\mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X})$ defined by:

$$\forall A, B \in \mathcal{P}(\mathcal{X}), \quad K_P(A, B) := \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in B} K(\mathbf{x}, \mathbf{y})$$

is a p.d. kernel on $\mathcal{P}(\mathcal{X})$.

LA kernel is p.d.: proof (6/11)

Proof of lemma

Let $\Phi : \mathcal{X} \mapsto \mathcal{H}$ be such that

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

Then, for $A, B \in \mathcal{P}(\mathcal{X})$, we get:

$$\begin{aligned} K_P(A, B) &= \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in B} \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}} \\ &= \left\langle \sum_{\mathbf{x} \in A} \Phi(\mathbf{x}), \sum_{\mathbf{y} \in B} \Phi(\mathbf{y}) \right\rangle_{\mathcal{H}} \\ &= \langle \Phi_P(A), \Phi_P(B) \rangle_{\mathcal{H}} , \end{aligned}$$

with $\Phi_P(A) := \sum_{\mathbf{x} \in A} \Phi(\mathbf{x})$. $\square$

LA kernel is p.d.: proof (7/11)

Definition: Convolution kernel (Haussler, 1999)

Let K_1 and K_2 be two p.d. kernels for strings. The convolution of K_1 and K_2, denoted $K_1 \star K_2$, is defined for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ by:

$$K_1 \star K_2(\mathbf{x}, \mathbf{y}) := \sum_{\mathbf{x}_1\mathbf{x}_2=\mathbf{x}, \mathbf{y}_1\mathbf{y}_2=\mathbf{y}} K_1(\mathbf{x}_1, \mathbf{y}_1) K_2(\mathbf{x}_2, \mathbf{y}_2).$$

Lemma

If K_1 and K_2 are p.d. then $K_1 \star K_2$ is p.d..

LA kernel is p.d.: proof (8/11)

Proof of lemma

Let $\mathcal{X}$ be the set of finite-length strings. For $\mathbf{x} \in \mathcal{X}$, let

$$R(\mathbf{x}) = \{(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{X} \times \mathcal{X} : \mathbf{x} = \mathbf{x}_1\mathbf{x}_2\} \subset \mathcal{X} \times \mathcal{X} .$$

We can then write

$$K_1 \star K_2(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x}_1,\mathbf{x}_2)\in R(\mathbf{x})} \sum_{(\mathbf{y}_1,\mathbf{y}_2)\in R(\mathbf{y})} K_1(\mathbf{x}_1, \mathbf{y}_1) K_2(\mathbf{x}_2, \mathbf{y}_2)$$

which is a p.d. kernel by the previous lemmas. $\square$

LA kernel is p.d.: proof (9/11)

3 basic string kernels

- The constant kernel:

$$K_0\left(\mathbf{x},\mathbf{y}\right) := 1\,.$$

- A kernel for letters:

$$K_a^{(\beta)}\left(\mathbf{x},\mathbf{y}\right) := \begin{cases} 0 & \text{if } \mid \mathbf{x}\mid \neq 1 \text{ where } \mid \mathbf{y}\mid \neq 1\,, \\ \exp\left(\beta S(\mathbf{x},\mathbf{y})\right) & \text{otherwise}\,. \end{cases}$$

- A kernel for gaps:

$$K_g^{(\beta)}\left(\mathbf{x},\mathbf{y}\right) = \exp\left[\beta\left(g\left(\mid \mathbf{x}\mid\right) + g\left(\mid \mathbf{y}\mid\right)\right)\right]\,.$$

LA kernel is p.d.: proof (10/11)

Remark

- $S : \mathcal{A}^2 \to \mathbb{R}$ is the similarity function between letters used in the alignment score. $K_a^{(\beta)}$ is only p.d. when the matrix:

  $$\left(\exp\left(\beta s(a,b)\right)\right)_{(a,b)\in\mathcal{A}^2}$$

  is positive semidefinite (this is true for all β when s is conditionally p.d..
- g is the gap penalty function used in alignment score. The gap kernel is always p.d. (with no restriction on g) because it can be written as:

  $$K_g^{(\beta)}(\mathbf{x},\mathbf{y}) = \exp\left(\beta g\left(\mid \mathbf{x} \mid\right)\right) \times \exp\left(\beta g\left(\mid \mathbf{y} \mid\right)\right) .$$

LA kernel is p.d.: proof (11/11)

Lemma

The local alignment kernel is a (limit) of convolution kernel:

$$K_{LA}^{(\beta)} = \sum_{n=0}^{\infty} K_0 \star \left(K_a^{(\beta)} \star K_g^{(\beta)} \right)^{(n-1)} \star K_a^{(\beta)} \star K_0 .$$

As such it is p.d..

Proof (sketch)

- By induction on n (simple but long to write).
- See details in Vert et al. (2004).

LA kernel computation

- We assume an affine gap penalty:

$$\begin{cases} g(0) & = 0, \\ g(n) & = d + e(n-1) \text{ si } n \geq 1, \end{cases}$$

- The LA kernel can then be computed by dynamic programming by:

$$K_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = 1 + X_2(|\mathbf{x}|,|\mathbf{y}|) + Y_2(|\mathbf{x}|,|\mathbf{y}|) + M(|\mathbf{x}|,|\mathbf{y}|),$$

  where $M(i,j), X(i,j), Y(i,j), X_2(i,j)$, and $Y_2(i,j)$ for $0 \leq i \leq |\mathbf{x}|$, and $0 \leq j \leq |\mathbf{y}|$ are defined recursively.

LA kernel is p.d.: proof (/)

Initialization

$$\begin{cases} M(i,0) = M(0,j) = 0, \\ X(i,0) = X(0,j) = 0, \\ Y(i,0) = Y(0,j) = 0, \\ X_2(i,0) = X_2(0,j) = 0, \\ Y_2(i,0) = Y_2(0,j) = 0, \end{cases}$$

LA kernel is p.d.: proof (/)

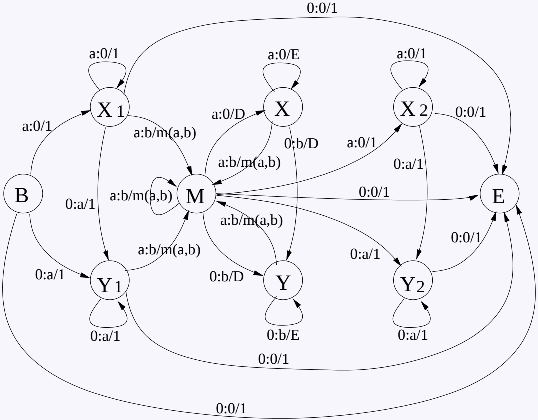
Recursion

For $i = 1, \ldots, |\mathbf{x}|$ and $j = 1, \ldots, |\mathbf{y}|$:

$$\begin{cases} M(i,j) & = \exp(\beta S(x_i, y_j)) \Big[1 + X(i-1, j-1) \\ & \qquad\qquad + Y(i-1, j-1) + M(i-1, j-1) \Big], \\ X(i,j) & = \exp(\beta d) M(i-1, j) + \exp(\beta e) X(i-1, j), \\ Y(i,j) & = \exp(\beta d) \left[M(i, j-1) + X(i, j-1) \right] \\ & \qquad\qquad + \exp(\beta e) Y(i, j-1), \\ X_2(i,j) & = M(i-1, j) + X_2(i-1, j), \\ Y_2(i,j) & = M(i, j-1) + X_2(i, j-1) + Y_2(i, j-1). \end{cases}$$

LA kernel in practice

- Implementation by a finite-state transducer in $O(|\mathbf{x}| \times |\mathbf{x}'|)$

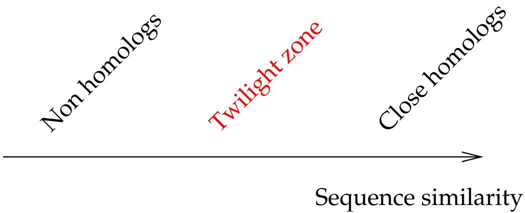


- In practice, values are too large (exponential scale) so taking its logarithm is a safer choice (but not p.d. anymore!)

Outline

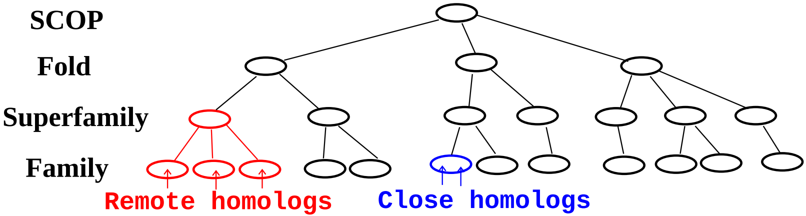
5. The Kernel Jungle
    - Green, Mercer, Herglotz, Bochner and friends
    - Kernels for probabilistic models
    - Kernels for biological sequences
        - Motivations and history of genomics
        - Kernels derived from large feature spaces
        - Kernels derived from generative models
        - Kernels derived from a similarity measure
        - Application to remote homology detection
    - Kernels for graphs
    - Kernels on graphs

Remote homology



- Homologs have common ancestors
- Structures and functions are more conserved than sequences
- Remote homologs can not be detected by direct sequence comparison

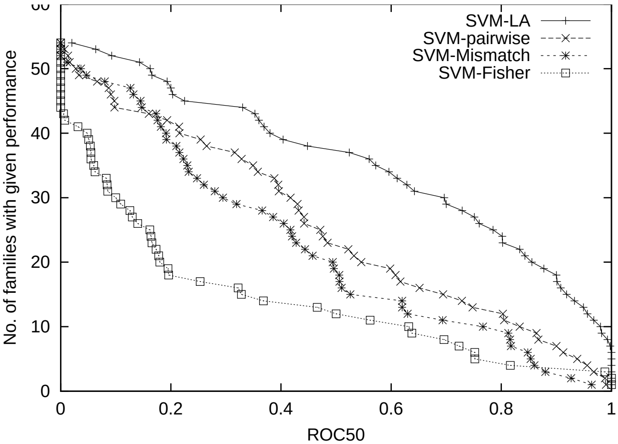
SCOP database



A benchmark experiment

- Goal: recognize directly the superfamily
- Training: for a sequence of interest, positive examples come from the same superfamily, but different families. Negative from other superfamilies.
- Test: predict the superfamily.

Difference in performance



Performance on the SCOP superfamily recognition benchmark (from Saigo et al., 2004).

String kernels: Summary

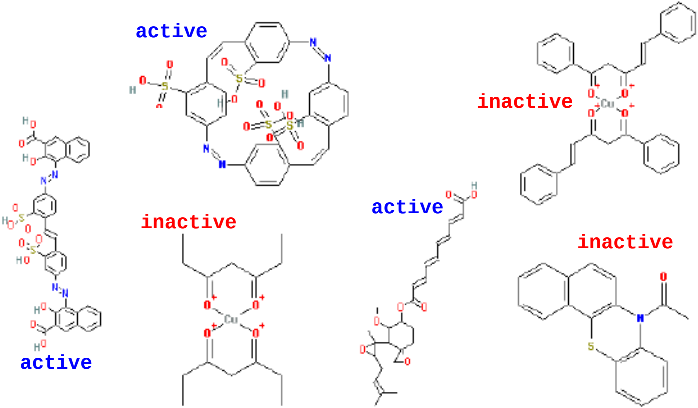
- A variety of principles for string kernel design have been proposed.
- Good kernel design is important for each data and each task. Performance is not the only criterion.
- Still an art, although principled ways have started to emerge.
- Fast implementation with string algorithms is often possible.
- Their application goes well beyond computational biology.

Outline

Outline

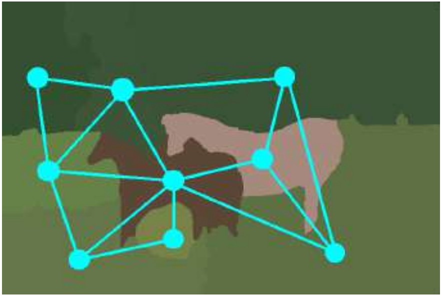
5. The Kernel Jungle
   - Green, Mercer, Herglotz, Bochner and friends
   - Kernels for probabilistic models
   - Kernels for biological sequences
   - Kernels for graphs
     - Motivation
     - Explicit enumeration of features
     - Challenges
     - Walk-based kernels
     - Applications
   - Kernels on graphs

Virtual screening for drug discovery



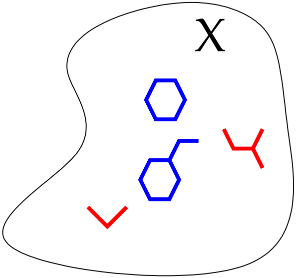
NCI AIDS screen results (from http://cactus.nci.nih.gov).

Image retrieval and classification



From Harchaoui and Bach (2007).

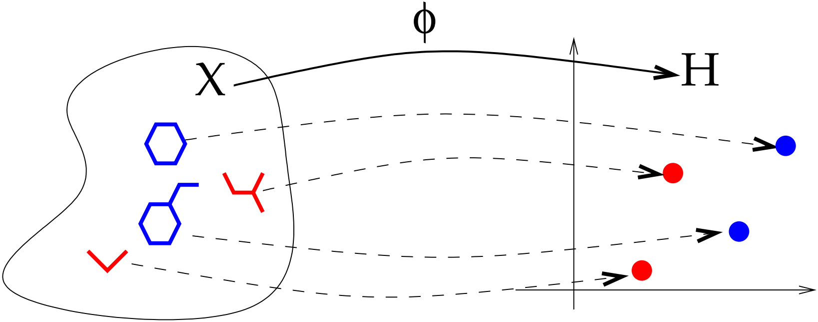
Our approach



Our approach

1. Represent each graph $\mathbf{x}$ in $\mathcal{X}$ by a vector $\Phi(\mathbf{x}) \in \mathcal{H}$, either explicitly or implicitly through the kernel

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}') \,.$$

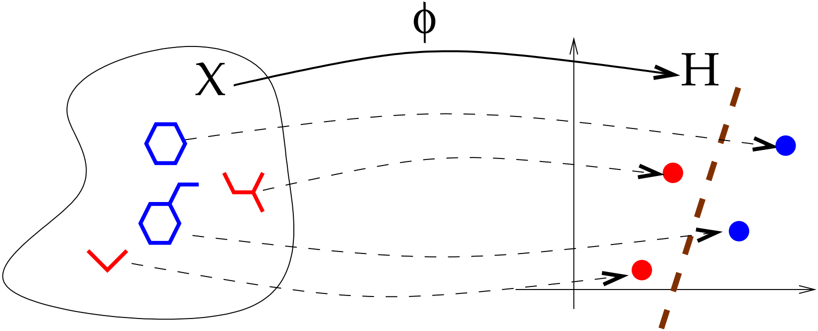


Our approach

1. Represent each graph $\mathbf{x}$ in $\mathcal{X}$ by a vector $\Phi(\mathbf{x}) \in \mathcal{H}$, either explicitly or implicitly through the kernel

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}') .$$

2. Use a linear method for classification in $\mathcal{H}$.

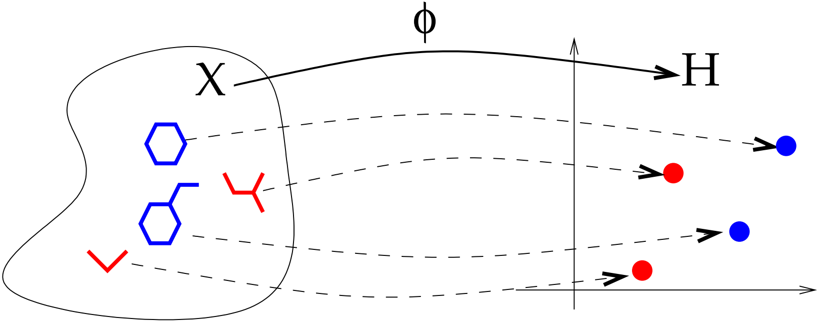


Outline

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

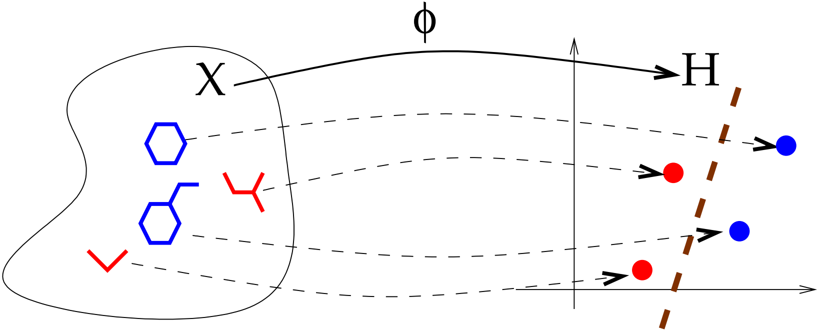
The approach

1. Represent explicitly each graph **x** by a vector of fixed dimension $\Phi(\mathbf{x}) \in \mathbb{R}^p$.



The approach

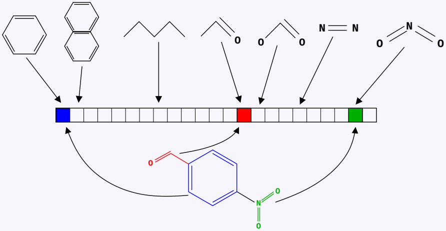
1. Represent explicitly each graph $\mathbf{x}$ by a vector of fixed dimension $\Phi(\mathbf{x}) \in \mathbb{R}^p$.
2. Use an algorithm for regression or pattern recognition in $\mathbb{R}^p$.



Example

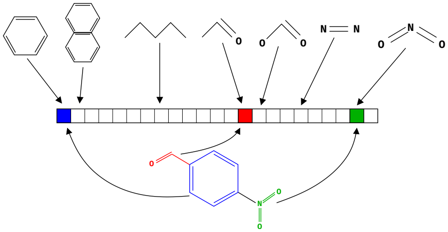
2D structural keys in chemoinformatics

- Index a molecule by a binary fingerprint defined by a limited set of predefined structures



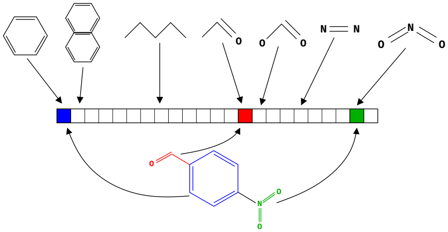
- Use a machine learning algorithm such as SVM, *k*NN, PLS, decision tree, etc.

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

Indexing by substructures

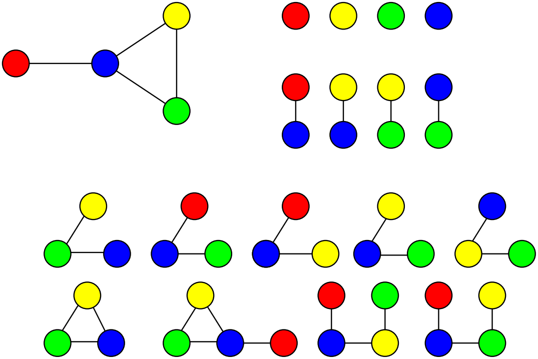


- Often we believe that the presence or absence of particular substructures may be important predictive patterns
- Hence it makes sense to represent a graph by features that indicate the presence (or the number of occurrences) of these substructures
- However, detecting the presence of particular substructures may be computationally challenging...

Subgraphs

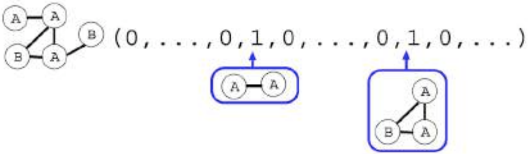
Definition

A subgraph of a graph (V, E) is a graph (V', E') with $V' \subset V$ and $E' \subset E$.

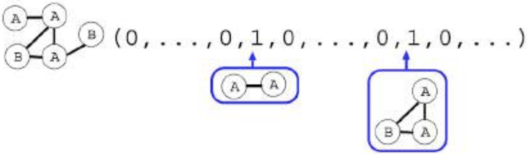


A graph and all its connected subgraphs.

Indexing by all subgraphs?



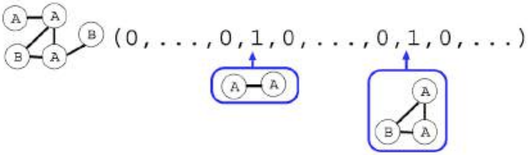
Indexing by all subgraphs?



Theorem

Computing all subgraph occurrences is NP-hard.

Indexing by all subgraphs?



Theorem

Computing all subgraph occurrences is NP-hard.

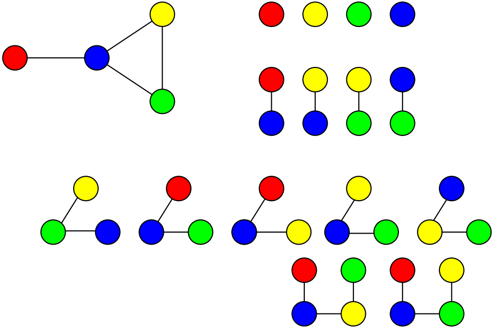
Proof

- The linear graph of size n is a subgraph of a graph X with n vertices iff X has a Hamiltonian path;
- The decision problem whether a graph has a Hamiltonian path is NP-complete.

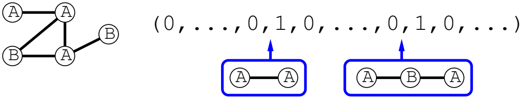
Paths

Definition

- A path of a graph (V, E) is 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.

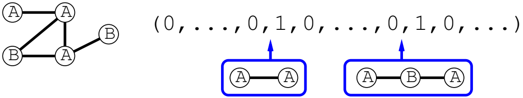


Indexing by all paths?



(0,...,0,1,0,...,0,1,0,...)

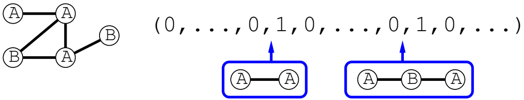
Indexing by all paths?



Theorem

Computing all path occurrences is NP-hard.

Indexing by all paths?



Theorem

Computing all path occurrences is NP-hard.

Proof

Same as for subgraphs.

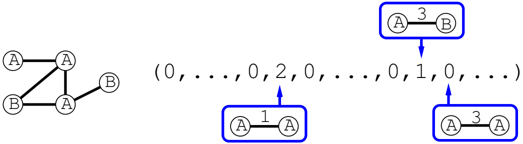
Indexing by what?

Substructure selection

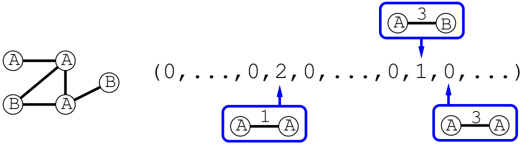
We can imagine more limited sets of substructures that lead to more computationnally efficient indexing (non-exhaustive list)

- substructures selected by domain knowledge (MDL fingerprint)
- all 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



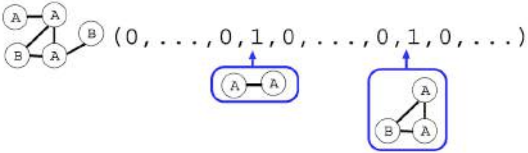
Example: Indexing by all shortest path lengths and their endpoint labels



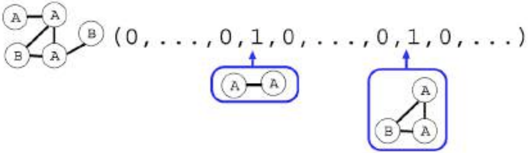
Properties (Borgwardt and Kriegel, 2005)

- 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



Example: Indexing by all subgraphs up to k vertices



Properties (Shervashidze et al., 2009)

- Naive enumeration scales as $O(n^k)$.
- Enumeration of connected graphlets in $O(nd^{k-1})$ for graphs with degree $\leq d$ and $k \leq 5$.
- Randomly sample subgraphs if enumeration is infeasible.

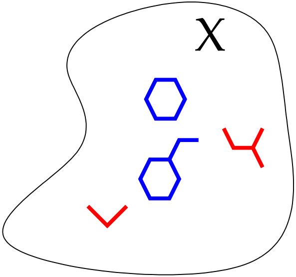
Summary

- Explicit computation of substructure occurrences can be computationnally 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

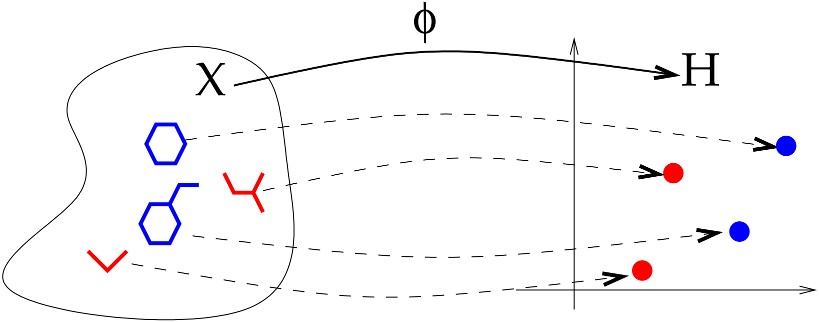
The idea



The idea

1. Represent implicitly each graph $\mathbf{x}$ in $\mathcal{X}$ by a vector $\Phi(\mathbf{x}) \in \mathcal{H}$ through the kernel

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}')\,.$$

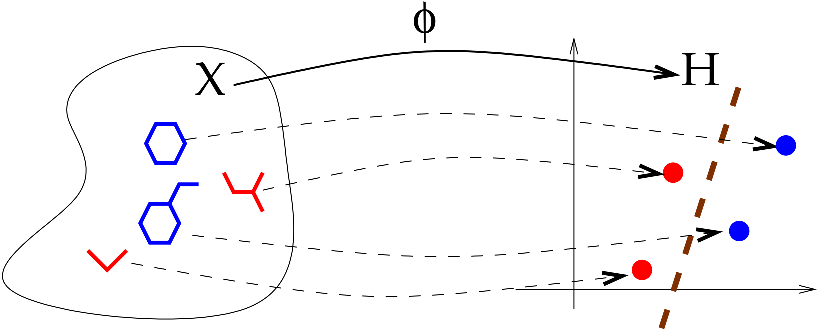


The idea

1. Represent implicitly each graph **x** in $\mathcal{X}$ by a vector $\Phi(\mathbf{x}) \in \mathcal{H}$ through the kernel

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}') .$$

2. Use a kernel method for classification in $\mathcal{H}$.



Expressiveness vs Complexity

Definition: Complete graph kernels

A graph kernel is complete if it 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

Proposition (Gärtner et al., 2003)

Computing any complete graph kernel is at least as hard as the graph isomorphism problem.

Complexity of complete kernels

Proposition (Gärtner et al., 2003)

Computing any complete graph kernel is at least as hard as the graph isomorphism problem.

Proof

- For any kernel K the complexity of computing d_K is the same as the complexity of computing K, because:

$$d_K(G_1, G_2)^2 = K(G_1, G_1) + K(G_2, G_2) - 2K(G_1, G_2)\,.$$

- If K is a complete graph kernel, then computing d_K solves the graph isomorphism problem ($d_K(G_1, G_2) = 0$ iff $G_1 \simeq G_2$). $\square$

Subgraph kernel

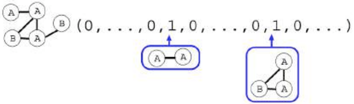
Definition

- Let $(\lambda_G)_{G\in\mathcal{X}}$ be a set or nonnegative real-valued weights
- For any graph $G \in \mathcal{X}$ and any connected graph $H \in \mathcal{X}$, let

$$\Phi_H(G) = \left| \left\{ G' \text{ is a subgraph of } G \ : \ G' \simeq H \right\} \right| .$$

- The subgraph kernel between any two graphs G_1 and $G_2 \in \mathcal{X}$ is defined by:

$$K_{subgraph}(G_1, G_2) = \sum_{\substack{H\in\mathcal{X} \\ H \text{ connected}}} \lambda_H \Phi_H(G_1)\Phi_H(G_2) .$$



Subgraph kernel complexity

Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is NP-hard.

Subgraph kernel complexity

Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is NP-hard.

Proof (1/2)

- Let P_n be the path graph with n vertices.
- Subgraphs of P_n are path graphs:

$$\Phi(P_n) = ne_{P_1} + (n-1)e_{P_2} + \ldots + e_{P_n}\,.$$

- 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 α_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 NP-hard.

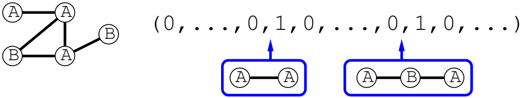
Proof (2/2)

- 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.,

$$\Phi(G)^\top \left(\sum_{i=1}^{n} \alpha_i \Phi(P_i) \right) = \sum_{i=1}^{n} \alpha_i K_{subgraph}(G, P_i) > 0 \,.$$

- The decision problem whether a graph has a Hamiltonian path is NP-complete. □

Path kernel



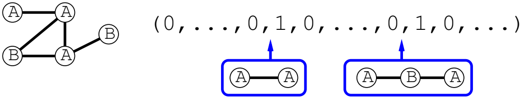
Definition

The path kernel is the subgraph kernel restricted to paths, i.e.,

$$K_{path}(G_1, G_2) = \sum_{H \in \mathcal{P}} \lambda_H \Phi_H(G_1) \Phi_H(G_2) \,,$$

where $\mathcal{P} \subset \mathcal{X}$ is the set of path graphs.

Path kernel



(0, . . . , 0, 1, 0, . . . , 0, 1, 0, . . .)

Definition

The path kernel is the subgraph kernel restricted to paths, i.e.,

$$K_{path}(G_1, G_2) = \sum_{H \in \mathcal{P}} \lambda_H \Phi_H(G_1) \Phi_H(G_2)\,,$$

where $\mathcal{P} \subset \mathcal{X}$ is the set of path graphs.

Proposition (Gärtner et al., 2003)

Computing the path kernel is NP-hard.

Summary

Expressiveness vs Complexity trade-off

- It is intractable to compute complete graph kernels.
- It is intractable to compute the subgraph kernels.
- Restricting subgraphs to be linear does not help: it is also intractable to compute the path kernel.
- One approach to define polynomial time computable graph kernels is to have the feature space be made up of graphs homomorphic to subgraphs, e.g., to consider walks instead of paths.

Outline

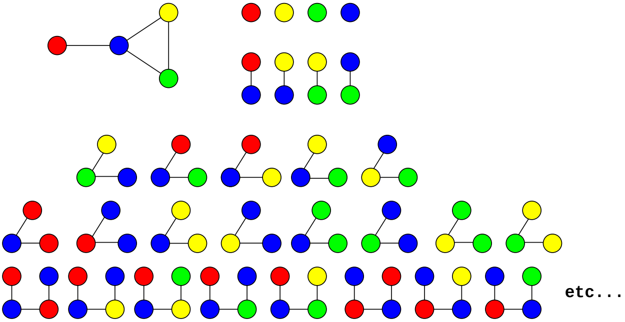
5 The Kernel Jungle

- 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

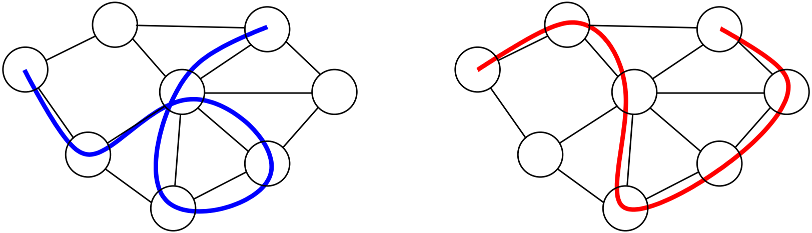
Walks

Definition

- A walk of a graph (V, E) is 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 $\mathcal{W}_n(G)$ the set of walks with n vertices of the graph G, and $\mathcal{W}(G)$ the set of all walks.



Walks $\neq$ paths



Walk kernel

Definition

- Let $\mathcal{S}_n$ denote the set of all possible label sequences of walks of length n (including vertex and edge labels), and $\mathcal{S} = \cup_{n \geq 1} \mathcal{S}_n$.
- For any graph $\mathcal{X}$ let a weight $\lambda_G(w)$ be associated to each walk $w \in \mathcal{W}(G)$.
- Let the feature vector $\Phi(G) = (\Phi_s(G))_{s \in \mathcal{S}}$ be defined by:

$$\Phi_s(G) = \sum_{w \in \mathcal{W}(G)} \lambda_G(w) \mathbf{1} \left(s \text{ is the label sequence of } w \right) .$$

Walk kernel

Definition

- Let $\mathcal{S}_n$ denote the set of all possible label sequences of walks of length n (including vertex and edge labels), and $\mathcal{S} = \cup_{n \geq 1} \mathcal{S}_n$.
- For any graph $\mathcal{X}$ let a weight $\lambda_G(w)$ be associated to each walk $w \in \mathcal{W}(G)$.
- Let the feature vector $\Phi(G) = (\Phi_s(G))_{s \in \mathcal{S}}$ be defined by:

$$\Phi_s(G) = \sum_{w \in \mathcal{W}(G)} \lambda_G(w) \mathbf{1} \left(s \text{ is the label sequence of } w \right) .$$

- A walk kernel is a graph kernel defined by:

$$K_{walk}(G_1, G_2) = \sum_{s \in \mathcal{S}} \Phi_s(G_1) \Phi_s(G_2) .$$

Walk kernel examples

Examples

- The 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 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 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).

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:

  $$K(G_1, G_2) = P(label(W_1) = label(W_2)),$$

  where W_1 and W_2 are two independent random walks on G_1 and G_2, respectively (Kashima et al., 2003).
- The geometric walk kernel is obtained (when it converges) with $\lambda_G(w) = \beta^{length(w)}$, for $\beta > 0$. In that case the feature space is of infinite dimension (Gärtner et al., 2003).

Computation of walk kernels

Proposition

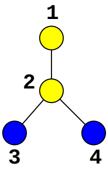
These three kernels (*n*th-order, random and geometric walk kernels) can be computed efficiently in polynomial time.

Product graph

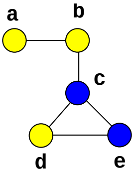
Definition

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs with labeled vertices. The product graph $G = G_1 \times G_2$ is the graph $G = (V, E)$ with:

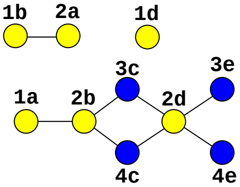
1. $V = \{(v_1, v_2) \in V_1 \times V_2 \; : \; v_1 \text{ and } v_2 \text{ have the same label}\}$,
2. $E = \{((v_1, v_2), (v'_1, v'_2)) \in V \times V \; : \; (v_1, v'_1) \in E_1 \text{ and } (v_2, v'_2) \in E_2\}$.



G1



G2



G1 x G2

Walk kernel and product graph

Lemma

There is a bijection between:

1. The pairs of walks $w_1 \in \mathcal{W}_n(G_1)$ and $w_2 \in \mathcal{W}_n(G_2)$ with the same label sequences,
2. The walks on the product graph $w \in \mathcal{W}_n(G_1 \times G_2)$.

Walk kernel and product graph

Lemma

There is a bijection between:

1. The pairs of walks $w_1 \in \mathcal{W}_n(G_1)$ and $w_2 \in \mathcal{W}_n(G_2)$ with the same label sequences,
2. The walks on the product graph $w \in \mathcal{W}_n(G_1 \times G_2)$.

Corollary

$$\begin{aligned} K_{walk}(G_1, G_2) &= \sum_{s \in \mathcal{S}} \Phi_s(G_1)\Phi_s(G_2) \\ &= \sum_{(w_1,w_2) \in \mathcal{W}(G_1) \times \mathcal{W}(G_1)} \lambda_{G_1}(w_1)\lambda_{G_2}(w_2)\mathbf{1}(l(w_1) = l(w_2)) \\ &= \sum_{w \in \mathcal{W}(G_1 \times G_2)} \lambda_{G_1 \times G_2}(w)\,. \end{aligned}$$

Computation of the nth-order walk kernel

- For the nth-order walk kernel we have $\lambda_{G_1 \times G_2}(w) = 1$ if the length of w is n, 0 otherwise.
- Therefore:

$$K_{nth\text{-}order}(G_1, G_2) = \sum_{w \in \mathcal{W}_n(G_1 \times G_2)} 1 \,.$$

- Let A be the adjacency matrix of $G_1 \times G_2$. Then we get:

$$K_{nth\text{-}order}(G_1, G_2) = \sum_{i,j} [A^n]_{i,j} = \mathbf{1}^\top A^n \mathbf{1} \,.$$

- Computation in $O(n|V_1||V_2|d_1 d_2)$, 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:

$$\lambda_G(v_1 \ldots v_n) = \lambda^i(v_1) \prod_{i=2}^{n} \lambda^t(v_{i-1}, v_i)\,.$$

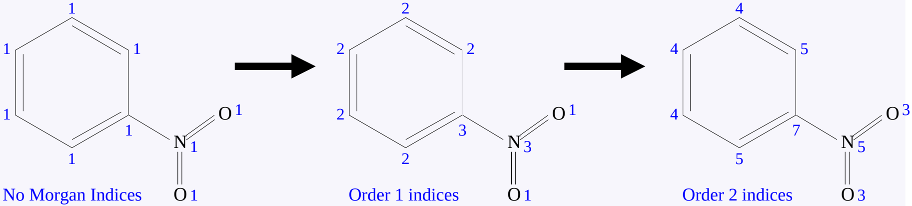
- Let Λ_i be the vector of $\lambda^i(v)$ and Λ_t be the matrix of $\lambda^t(v, v')$:

$$\begin{aligned} K_{walk}(G_1, G_2) &= \sum_{n=1}^{\infty} \sum_{w \in \mathcal{W}_n(G_1 \times G_2)} \lambda^i(v_1) \prod_{i=2}^{n} \lambda^t(v_{i-1}, v_i) \\ &= \sum_{n=0}^{\infty} \Lambda_i \Lambda_t^n \mathbf{1} \\ &= \Lambda_i \left(I - \Lambda_t\right)^{-1} \mathbf{1} \end{aligned}$$

- Computation in $O(|V_1|^3 |V_2|^3)$.

Extensions 1: Label enrichment

Atom relabeling with the Morgan index (Mahé et al., 2004)

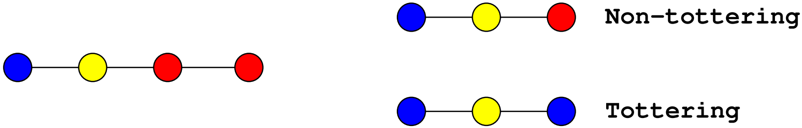


- Compromise between fingerprints and structural keys.
- Other relabeling schemes are possible.
- Faster computation with more labels (less matches implies a smaller product graph).

Extension 2: Non-tottering walk kernel

Tottering walks

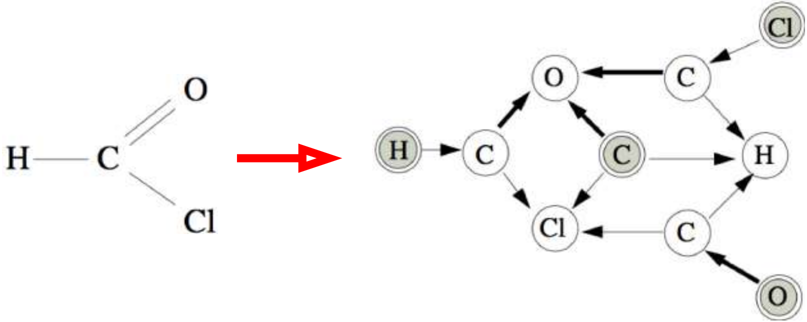
A tottering walk is a walk $w = v_1 \dots v_n$ with $v_i = v_{i+2}$ for some i.



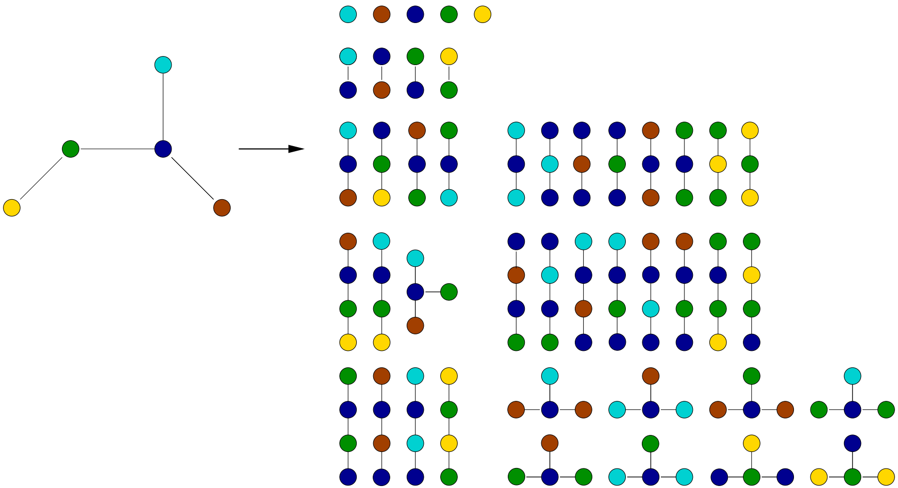
- Tottering walks seem irrelevant for many applications.
- Focusing on non-tottering walks is a way to get closer to the path kernel (e.g., equivalent on trees).

Computation of the non-tottering walk kernel (Mahé et al., 2005)

- Second-order Markov random walk to prevent tottering walks
- Written as a first-order Markov random walk on an augmented graph
- Normal walk kernel on the augmented graph (which is always a directed graph).

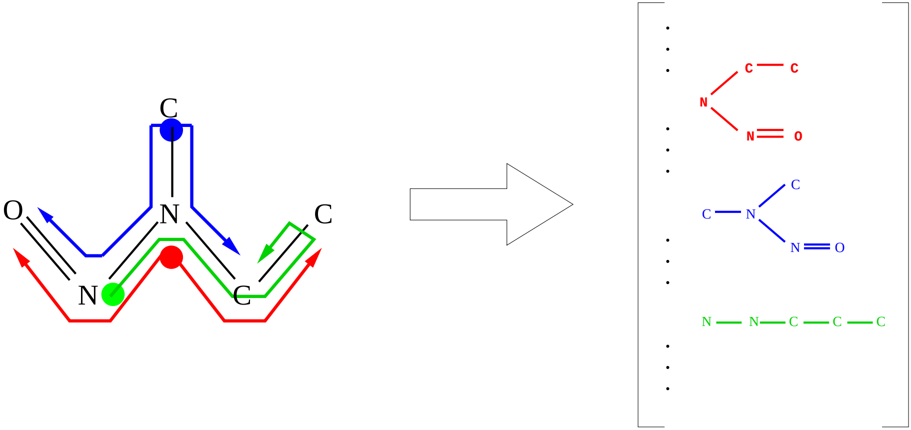


Extension 3: Subtree kernels



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 $\mathcal{T}(v, n)$ denotes the weighted number of subtrees of depth n rooted at the vertex v, then:

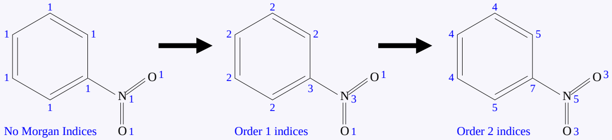
$$\mathcal{T}(v, n+1) = \sum_{R \subset \mathcal{N}(v)} \prod_{v' \in R} \lambda_t(v, v') \mathcal{T}(v', n),$$

  where $\mathcal{N}(v)$ is the set of neighbors of v.
- Can be combined with the non-tottering graph transformation as preprocessing to obtain the non-tottering subtree kernel.

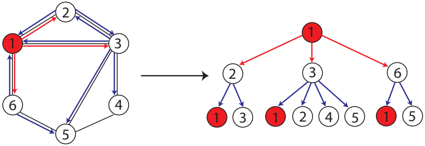
Back to label enrichment

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*.

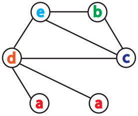


A full subtree pattern of order 2 rooted at node 1.

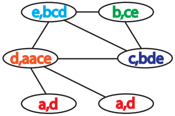
Label enrichment via the Weisfeiler-Lehman algorithm

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

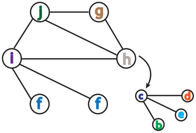


2. Label compression



a,d ⟶ f
b,ce ⟶ g
c,bde ⟶ h
d,aace ⟶ i
e,bcd ⟶ j

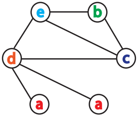
3. Relabeling



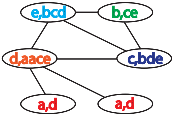
Label enrichment via the Weisfeiler-Lehman algorithm

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

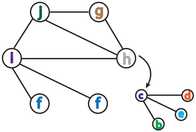


2. Label compression



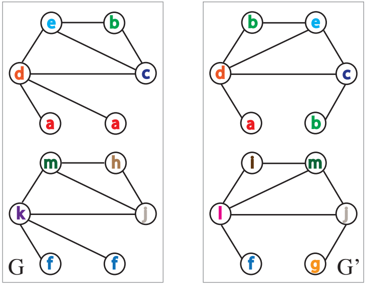
a,d ⟶ f
b,ce ⟶ g
c,bde ⟶ h
d,aace ⟶ i
e,bcd ⟶ j

3. Relabeling



Compressed labels represent full subtree patterns.

Weisfeiler-Lehman (WL) subtree kernel



$$\varphi^{(1)}_{WLsubtree}(G) = (\underset{a}{2}, \underset{b}{1}, \underset{c}{1}, \underset{d}{1}, \underset{e}{1}, \underset{f}{2}, \underset{g}{0}, \underset{h}{1}, \underset{i}{0}, \underset{j}{1}, \underset{k}{1}, \underset{l}{0}, \underset{m}{1})$$

$$\varphi^{(1)}_{WLsubtree}(G') = (\underbrace{\underset{a}{1}, \underset{b}{2}, \underset{c}{1}, \underset{d}{1}, \underset{e}{1}}_{\text{Counts of original node labels}}, \underbrace{\underset{f}{1}, \underset{g}{1}, \underset{h}{0}, \underset{i}{1}, \underset{j}{1}, \underset{k}{0}, \underset{l}{1}, \underset{m}{1}}_{\text{Counts of compressed node labels}})$$

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).

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

Application in chemoinformatics (Mahé et al., 2005)

MUTAG dataset

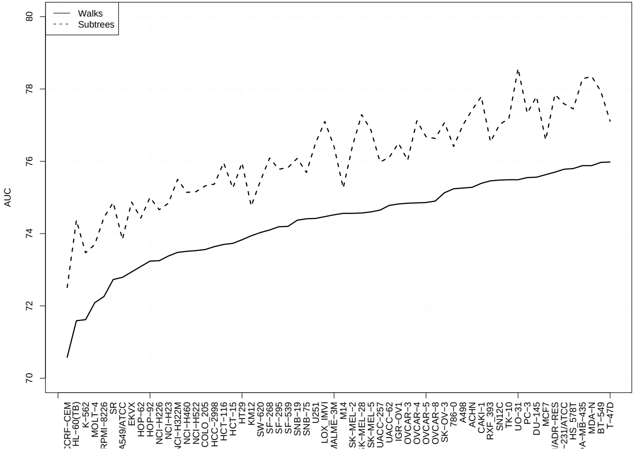
- aromatic/hetero-aromatic compounds
- high mutagenic activity /no mutagenic activity, assayed in *Salmonella typhimurium*.
- 188 compounds: 125 + / 63 -

Results

10-fold cross-validation accuracy

Method	Accuracy
Progol1	81.4%
2D kernel	91.2%

2D subtree vs walk kernels



Screening of inhibitors for 60 cancer cell lines.

Comparison of several graph feature extraction methods/kernels (Shervashidze et al., 2011)

10-fold cross-validation accuracy on garph classification problems in chemo- and bioinformatics:

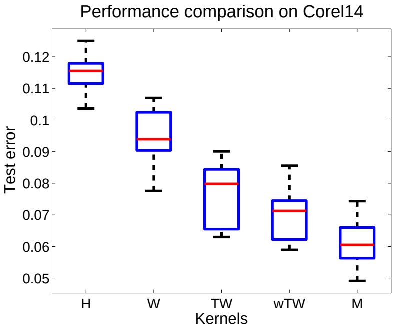
- NCI1 and NCI109 - active/inactive compounds in an anti-cancer screen
- ENZYMES - 6 types of enzymes from the BRENDA database

Method/Data Set	NCI1	NCI109	ENZYMES
WL subtree	82.19 (± 0.18)	82.46 (±0.24)	52.22 (±1.26)
WL shortest path	84.55 (±0.36)	83.53 (±0.30)	59.05 (±1.05)
Ramon & Gärtner	61.86 (±0.27)	61.67 (±0.21)	13.35 (±0.87)
Geometric *p*-walk	58.66 (±0.28)	58.36 (±0.94)	27.67 (±0.95)
Geometric walk	64.34 (±0.27)	63.51 (± 0.18)	21.68 (±0.94)
Graphlet count	66.00 (±0.07)	66.59 (±0.08)	32.70 (±1.20)
Shortest path	73.47 (±0.11)	73.07 (±0.11)	41.68 (±1.79)

Image classification (Harchaoui and Bach, 2007)

COREL14 dataset

- 1400 natural images in 14 classes
- Compare kernel between histograms (H), walk kernel (W), subtree kernel (TW), weighted subtree kernel (wTW), and a combination (M).



Summary: graph kernels

What we saw

- Kernels do not allow to overcome the NP-hardness of subgraph patterns.
- They allow to work with approximate subgraphs (walks, subtrees) in infinite dimension, thanks to the kernel trick.
- However: using kernels makes it difficult to come back to patterns after the learning stage.

Outline

Outline

5 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

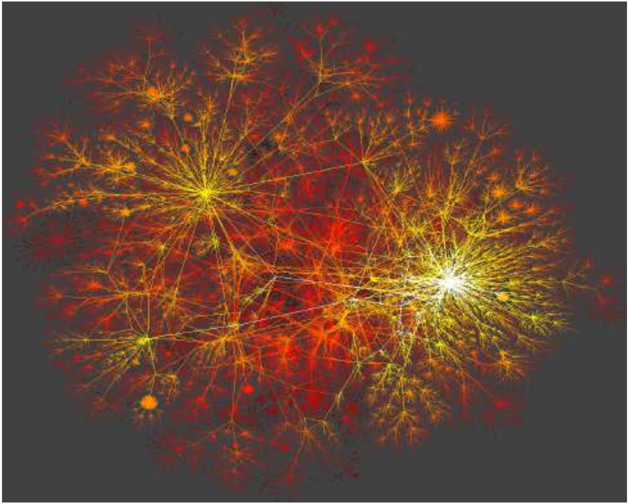
Graphs

Motivation

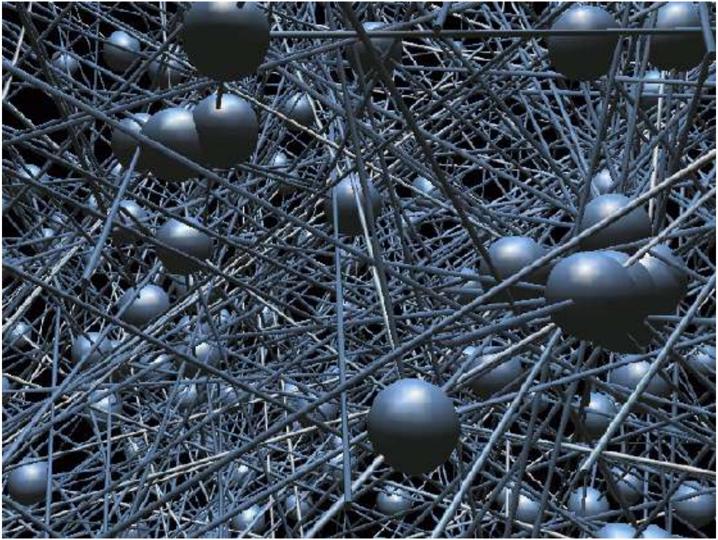
Data often come in the form of nodes in a graph for different reasons:

- by definition (interaction network, internet...)
- by discretization/sampling of a continuous domain
- by convenience (e.g., if only a similarity function is available)

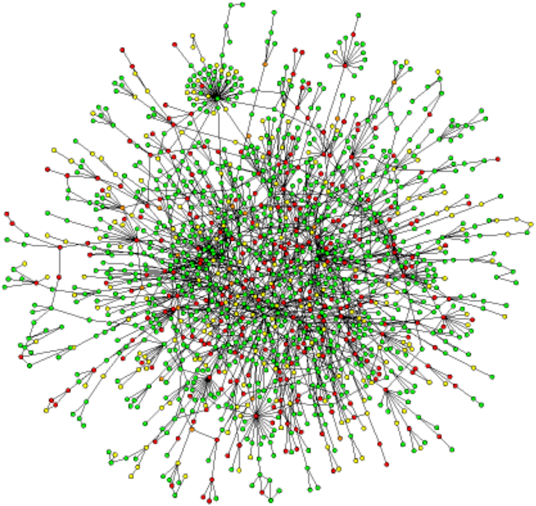
Example: web



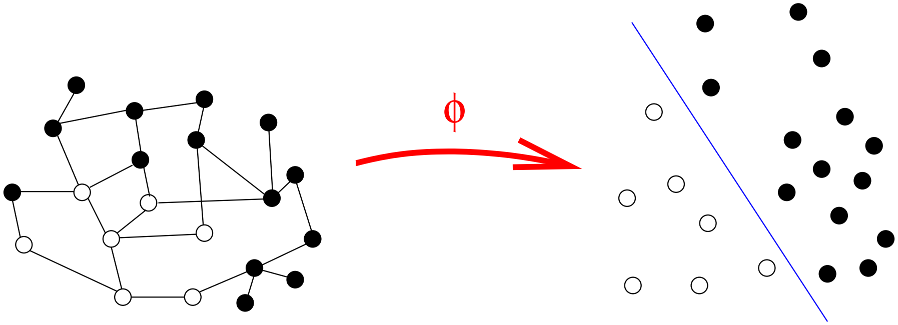
Example: social network



Example: protein-protein interaction



Kernel on a graph



- We need a kernel $K(\mathbf{x}, \mathbf{x}')$ 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

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 $\mathcal{X}$.
- How to "translate" the graph topology into the kernel?
  - Direct geometric approach: $K_{i,j}$ should be "large" when $\mathbf{x}_i$ and $\mathbf{x}_j$ are "close" to each other on the graph?
  - Functional approach: $\| f \|_K$ should be "small" when f is "smooth" on the graph?
  - Link discrete/continuous: is there an equivalent to the continuous Gaussian kernel on the graph (e.g., limit by fine discretization)?

Outline

5 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

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

- It defines a Hilbert distance:

$$d_K\left(\mathbf{x},\mathbf{x}'\right)^2 = K\left(\mathbf{x},\mathbf{x}\right) + K\left(\mathbf{x}',\mathbf{x}'\right) - 2K\left(\mathbf{x},\mathbf{x}'\right) .$$

- $-d_K^2$ is conditionally positive definite (c.p.d.), i.e.:

$$\forall t > 0 , \quad \exp\left(-t d_K\left(\mathbf{x},\mathbf{x}'\right)^2\right) \text{ is p.d.}$$

Example

A direct approach

- For $\mathcal{X} = \mathbb{R}^n$, the inner product is p.d.:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}' \,.$$

- The corresponding Hilbert distance is the Euclidean distance:

$$d_K \left(\mathbf{x}, \mathbf{x}'\right)^2 = \mathbf{x}^\top \mathbf{x} + \mathbf{x}'^\top \mathbf{x}' - 2\mathbf{x} \top \mathbf{x}' = ||\mathbf{x} - \mathbf{x}'||^2 \,.$$

- $-d_K^2$ is conditionally positive definite (c.p.d.), i.e.:

$$\forall t > 0 \,, \quad \exp\left(-t||\mathbf{x} - \mathbf{x}'||^2\right) \text{ is p.d.}$$

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

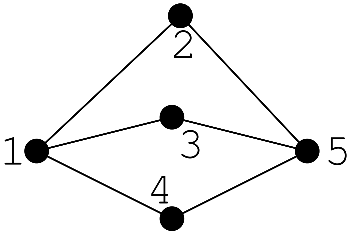
Graph embedding in a Hilbert space

- 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



$$d_G = \begin{pmatrix} 0 & 1 & 1 & 1 & 2 \\ 1 & 0 & 2 & 2 & 1 \\ 1 & 2 & 0 & 2 & 1 \\ 1 & 2 & 2 & 0 & 1 \\ 2 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\lambda_{\min}\left(\left[e^{(-0.2 d_G(i,j))}\right]\right) = -0.028 < 0\,.$$

Graph distances on trees are c.p.d.

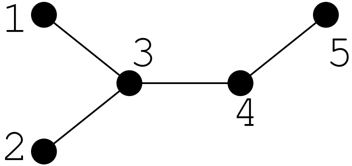
Proof

- 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 $\exp(-t d_G(x, x'))$ is p.d. for all $t > 0$.

Example

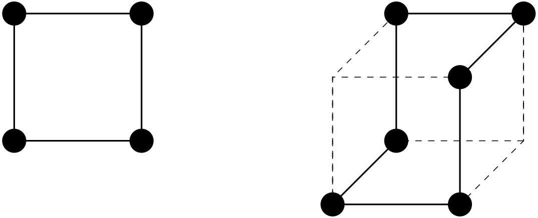


$$\left[e^{-d_G(i,j)}\right] = \begin{pmatrix} 1 & 0.14 & 0.37 & 0.14 & 0.05 \\ 0.14 & 1 & 0.37 & 0.14 & 0.05 \\ 0.37 & 0.37 & 1 & 0.37 & 0.14 \\ 0.14 & 0.14 & 0.37 & 1 & 0.37 \\ 0.05 & 0.05 & 0.14 & 0.37 & 1 \end{pmatrix}$$

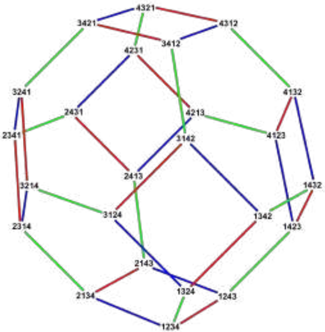
Graph distances on closed chains are c.p.d.

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 $e_1, \dots, e_p, -e_1, \dots, -e_p$ (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.



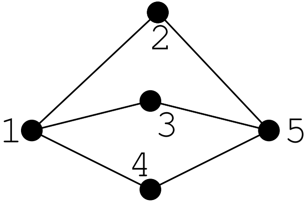
Another interesting graph

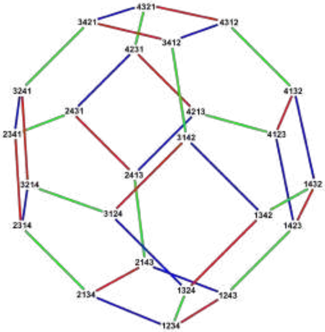


Cayley graph of $\mathbb{S}_4$

- Let $\mathbb{S}_n$ 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 $O(n \log n)$ *how?*
- d_G is c.p.d. *why?*
- See Jiao and Vert (2017)

Summary on graph distance





- 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

5. 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 $f : \mathcal{X} \to \mathbb{R}$;
- Show that it defines an RKHS and identify the corresponding kernel.

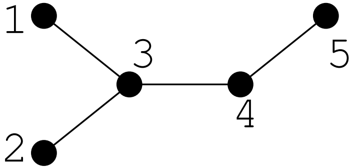
Notations

- $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ is finite.
- For $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, we note $\mathbf{x} \sim \mathbf{x}'$ to indicate the existence of an edge between $\mathbf{x}$ and $\mathbf{x}'$
- We assume that there is no self-loop $\mathbf{x} \sim \mathbf{x}$, and that there is a single connected component.
- The adjacency matrix is $A \in \mathbb{R}^{m \times m}$:

$$A_{i,j} = \begin{cases} 1 & \text{if } i \sim j, \\ 0 & \text{otherwise.} \end{cases}$$

- D is the diagonal matrix where $D_{i,i}$ is the number of neighbors of $\mathbf{x}_i$ ($D_{i,i} = \sum_{i=1}^{m} A_{i,j}$).

Example

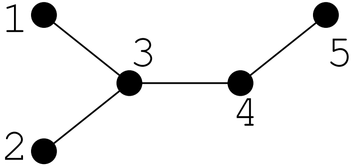


$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Graph Laplacian

Definition

The Laplacian of the graph is the matrix $L = D - A$.



$$L = D - A = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Properties of the Laplacian

Lemma

Let $L = D - A$ be the Laplacian of a connected graph:

- *For any $f : \mathcal{X} \to \mathbb{R}$,*

$$\Omega(f) := \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = f^\top L f$$

- *L is a symmetric positive semi-definite matrix*
- *0 is an eigenvalue with multiplicity 1 associated to the constant eigenvector $\mathbf{1} = (1, \ldots, 1)$*
- *The image of L is*

$$Im(L) = \left\{ f \in \mathbb{R}^m : \sum_{i=1}^{m} f_i = 0 \right\}$$

Proof: link between $\Omega(f)$ and L

$$
\begin{aligned}
\Omega\left(f\right) &= \sum_{i\sim j}\left(f\left(\mathbf{x}_i\right) - f\left(\mathbf{x}_j\right)\right)^2 \\
&= \sum_{i\sim j}\Big(f\left(\mathbf{x}_i\right)^2 + f\left(\mathbf{x}_j\right)^2 - 2f\left(\mathbf{x}_i\right)f\left(\mathbf{x}_j\right)\Big) \\
&= \sum_{i=1}^{m} D_{i,i} f\left(\mathbf{x}_i\right)^2 - 2\sum_{i\sim j} f\left(\mathbf{x}_i\right)f\left(\mathbf{x}_j\right) \\
&= f^\top D f - f^\top A f \\
&= f^\top L f
\end{aligned}
$$

Proof: eigenstructure of L

- L is symmetric because A and D are symmetric.
- For any $f \in \mathbb{R}^m$, $f^\top Lf = \Omega(f) \geq 0$, therefore the (real-valued) eigenvalues of L are ≥ 0 : L is therefore positive semi-definite.
- f is an eigenvector associated to eigenvalue 0
  iff $f^\top Lf = 0$
  iff $\sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = 0$,
  iff $f(\mathbf{x}_i) = f(\mathbf{x}_j)$ when $i \sim j$,
  iff f is constant (because the graph is connected).
- L being symmetric, $Im(L)$ is the orthogonal supplement of $Ker(L)$, that is, the set of functions orthogonal to $\mathbf{1}$. □

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*

$$\Omega(f) = \sum_{i \sim j} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2$$

is a RKHS whose reproducing kernel is L^**, the pseudo-inverse of the graph Laplacian.*

In case of...

Pseudo-inverse of L

Remember the pseudo-inverse L^* of L is the linear application that is equal to:

- 0 on $Ker(L)$
- L^{-1} on $Im(L)$, 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 $L^* L = L L^* = \Pi_{\mathcal{H}}$, the projection onto $Im(L) = \mathcal{H}$.

Proof (1/2)

- Resticted 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 $\mathcal{H} = Im(L)$). 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:

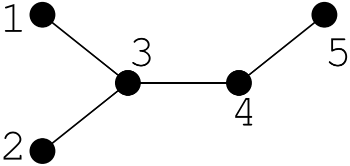
$$\begin{cases} \forall \mathbf{x} \in \mathcal{X}, & K_{\mathbf{x}} \in \mathcal{H} \ , \\ \forall (\mathbf{x}, f) \in \mathcal{X} \times \mathcal{H}, & \langle f, K_{\mathbf{x}} \rangle = f(\mathbf{x}) \ . \end{cases}$$

- $Ker(K) = Ker(L^*) = Ker(L)$, implying $K\mathbf{1} = 0$. Therefore, each row/column of K is in $\mathcal{H}$.
- For any $f \in \mathcal{H}$, if we note $g_i = \langle K(i, \cdot), f \rangle$ we get:

$$g = KLf = L^*Lf = \Pi_{\mathcal{H}}(f) = f \ .$$

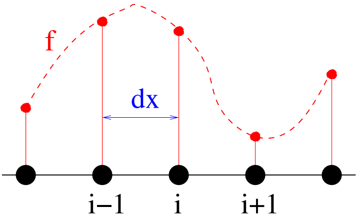
As a conclusion $K = L^*$ is the reproducing kernel of $\mathcal{H}$. $\square$

Example



$$L^* = \begin{pmatrix} 0.88 & -0.12 & 0.08 & -0.32 & -0.52 \\ -0.12 & 0.88 & 0.08 & -0.32 & -0.52 \\ 0.08 & 0.08 & 0.28 & -0.12 & -0.32 \\ -0.32 & -0.32 & -0.12 & 0.48 & 0.28 \\ -0.52 & -0.52 & -0.32 & 0.28 & 1.08 \end{pmatrix}$$

Interpretation of the Laplacian



$$\begin{aligned}\Delta f(x) &= f''(x)\\ &\sim \frac{f'(x+dx/2)-f'(x-dx/2)}{dx}\\ &\sim \frac{f(x+dx)-f(x)-f(x)+f(x-dx)}{dx^2}\\ &= \frac{f_{i-1}+f_{i+1}-2f(x)}{dx^2}\\ &= -\frac{Lf(i)}{dx^2}.\end{aligned}$$

Interpretation of regularization

For $f = [0,1] \to \mathbb{R}$ and $x_i = i/m$, we have:

$$\begin{aligned}
\Omega(f) &= \sum_{i=1}^{m} \left(f\left(\frac{i+1}{m}\right) - f\left(\frac{i}{m}\right) \right)^2 \\
&\sim \sum_{i=1}^{m} \left(\frac{1}{m} \times f'\left(\frac{i}{m}\right) \right)^2 \\
&= \frac{1}{m} \times \frac{1}{m} \sum_{i=1}^{m} f'\left(\frac{i}{m}\right)^2 \\
&\sim \frac{1}{m} \int_0^1 f'(t)^2 dt.
\end{aligned}$$

Outline

5. 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 $\mathbb{R}^d$:

$$K_t\left(\mathbf{x},\mathbf{x}'\right) = \frac{1}{(4\pi t)^{\frac{d}{2}}} \exp\left(-\frac{\parallel \mathbf{x}-\mathbf{x}' \parallel^2}{4t}\right).$$

- 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 $\mathbf{x}_0 \in \mathbb{R}^d$*, the function:*

$$K_{\mathbf{x}_0}(\mathbf{x}, t) = K_t(\mathbf{x}_0, \mathbf{x}) = \frac{1}{(4\pi t)^{\frac{d}{2}}} \exp\left(-\frac{\| \mathbf{x} - \mathbf{x}_0 \|^2}{4t}\right)$$

is solution of the diffusion equation:

$$\frac{\partial}{\partial t} K_{\mathbf{x}_0}(\mathbf{x}, t) = \Delta K_{\mathbf{x}_0}(\mathbf{x}, t)$$

with initial condition $K_{\mathbf{x}_0}(\mathbf{x}, 0) = \delta_{\mathbf{x}_0}(\mathbf{x})$

(proof by direct computation).

Discrete diffusion equation

For finite-dimensional $f_t \in \mathbb{R}^m$, the diffusion equation becomes:

$$\frac{\partial}{\partial t} f_t = -L f_t$$

which admits the following solution:

$$f_t = f_0 e^{-tL}$$

with

$$e^{-tL} = I - tL + \frac{t^2}{2!} L^2 - \frac{t^3}{3!} L^3 + \dots$$

Diffusion kernel (Kondor and Lafferty, 2002)

This suggest to consider:

$$K = e^{-tL}$$

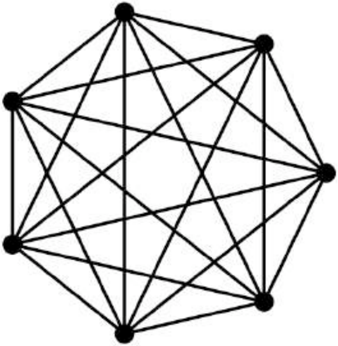
which is indeed symmetric positive semi-definite because if we write:

$$L = \sum_{i=1}^{m} \lambda_i u_i u_i^\top \quad (\lambda_i \geq 0)$$

we obtain:

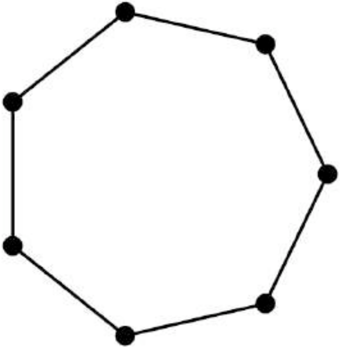
$$K = e^{-tL} = \sum_{i=1}^{m} e^{-t\lambda_i} u_i u_i^\top$$

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



$$K_{i,j} = \frac{1}{m} \sum_{\nu=0}^{m-1} \exp\left[-2t\left(1 - \cos\frac{2\pi\nu}{m}\right)\right] \cos\frac{2\pi\nu(i-j)}{m}.$$

Outline

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 $\int \left| \hat{f}(\omega) \right|^2 e^{\sigma^2\omega^2} d\omega$ ...

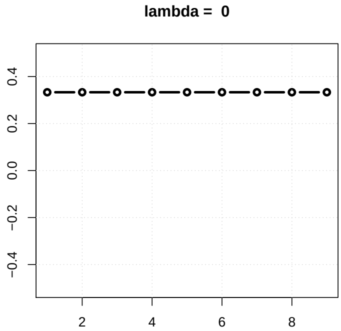
Discrete Fourier transform

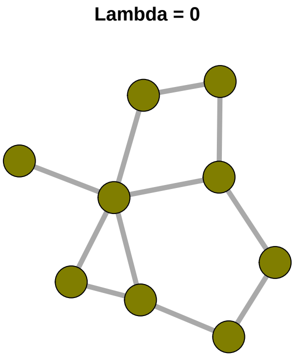
Definition

The vector $\hat{f} = \left(\hat{f}_1, \ldots, \hat{f}_m\right)^\top$ is called the discrete Fourier transform of $f \in \mathbb{R}^n$

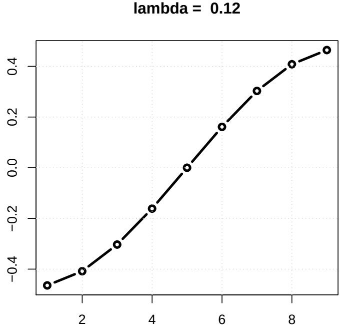
- The eigenvectors of the Laplacian are the discrete equivalent to the sine/cosine Fourier basis on $\mathbb{R}^n$.
- The eigenvalues λ_i are the equivalent to the frequencies ω^2
- Successive eigenvectors “oscillate” increasingly as eigenvalues get more and more negative.

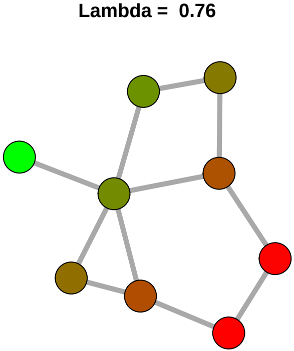
Examples



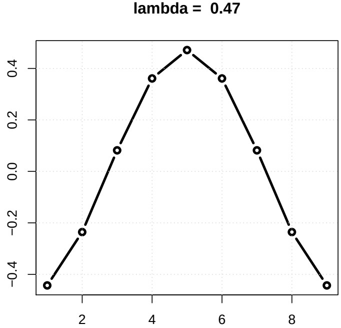


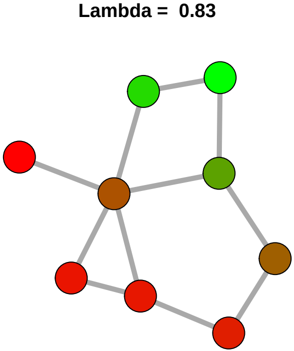
Examples



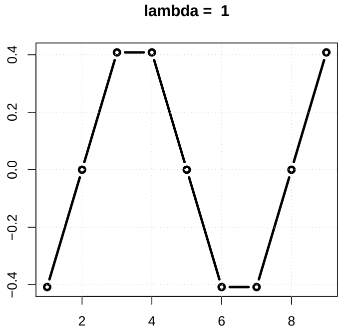


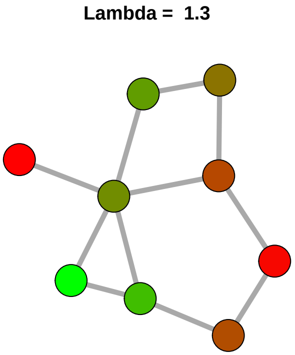
Examples



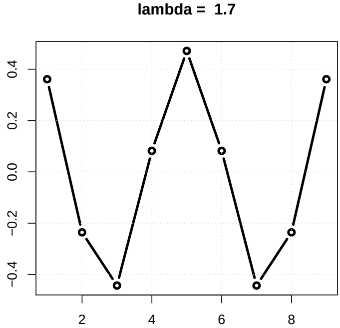


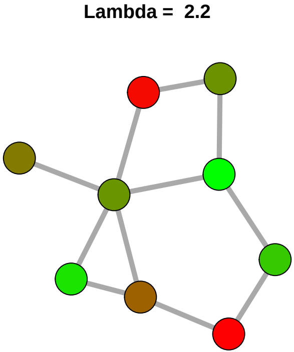
Examples



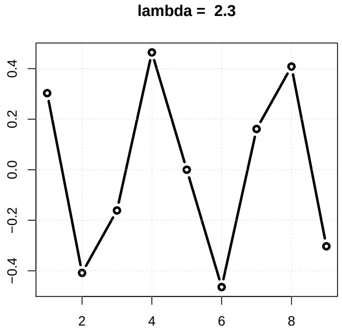


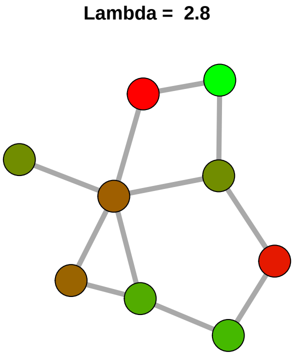
Examples


lambda = 1.7

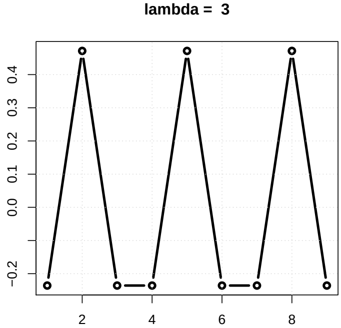

Lambda = 2.2

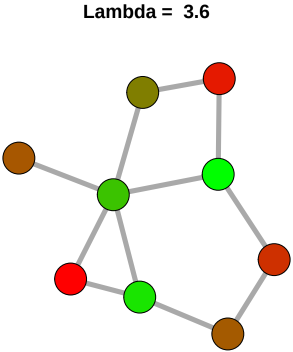
Examples



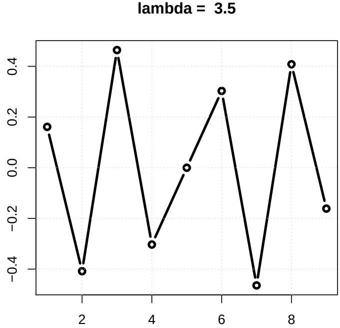


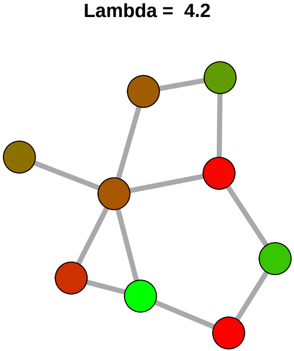
Examples



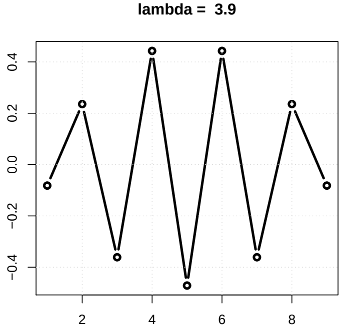


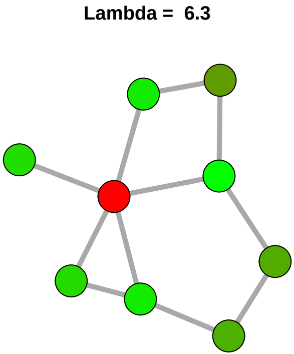
Examples





Examples


lambda = 3.9


Lambda = 6.3

Generalization

This observation suggests to define a whole family of kernels:

$$K_r = \sum_{i=1}^{m} r(\lambda_i) u_i u_i^\top$$

associated with the following RKHS norms:

$$\| f \|_{K_r}^2 = \sum_{i=1}^{m} \frac{\hat{f}_i^2}{r(\lambda_i)}$$

where $r : \mathbb{R}^+ \to \mathbb{R}_*^+$ is a non-increasing function.

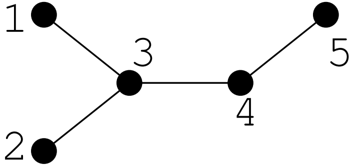
Example : regularized Laplacian

$$r(\lambda) = \frac{1}{\lambda + \epsilon}, \qquad \epsilon > 0$$

$$K = \sum_{i=1}^{m} \frac{1}{\lambda_i + \epsilon} u_i u_i^\top = (L + \epsilon I)^{-1}$$

$$\| f \|_K^2 = f^\top K^{-1} f = \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 + \epsilon \sum_{i=1}^{m} f(\mathbf{x}_i)^2 .$$

Example



$$(L+I)^{-1} = \begin{pmatrix} 0.60 & 0.10 & 0.19 & 0.08 & 0.04 \\ 0.10 & 0.60 & 0.19 & 0.08 & 0.04 \\ 0.19 & 0.19 & 0.38 & 0.15 & 0.08 \\ 0.08 & 0.08 & 0.15 & 0.46 & 0.23 \\ 0.04 & 0.04 & 0.08 & 0.23 & 0.62 \end{pmatrix}$$

Outline

5 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

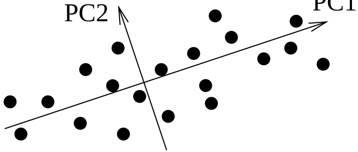
- A classical relaxation of graph partitioning is:

$$\min_{f\in\mathbb{R}^{\mathcal{X}}} \sum_{i\sim j} (f_i - f_j)^2 \quad \text{s.t.} \sum_i f_i^2 = 1$$

- This can be rewritten

$$\max_f \sum_i f_i^2 \text{ s.t. } \quad \| f \|_{\mathcal{H}} \leq 1$$

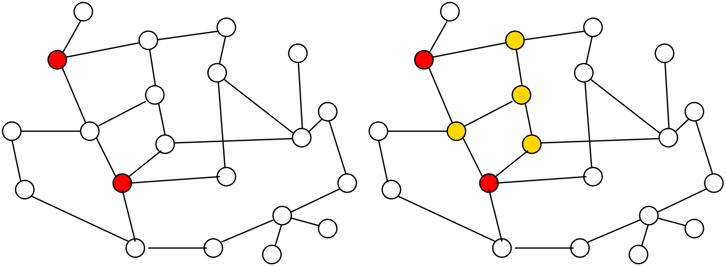
- This is principal component analysis in the RKHS ("kernel PCA")



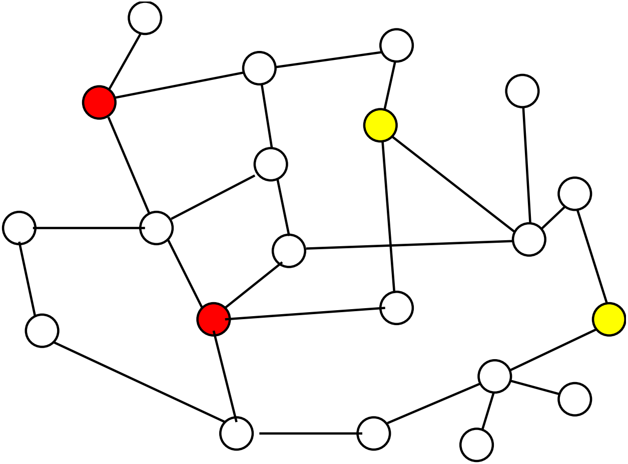
Applications 2: search on a graph

- Let $x_1, \ldots, x_q$ be a set of q nodes (the query). How to find "similar" nodes (and rank them)?
- One solution:

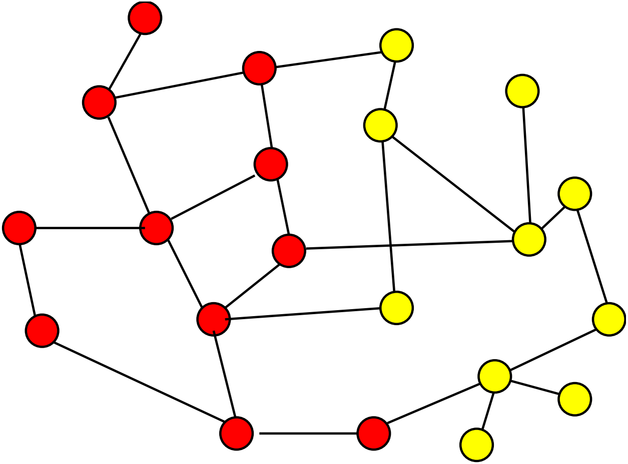
$$\min_f \| f \|_{\mathcal{H}} \quad \text{s.t.} \quad f(x_i) \geq 1 \text{ for } i = 1, \ldots, q.$$



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 β_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 β, e.g.:

- Euclidean norm (support vector machines, ridge regression): $\parallel \beta \parallel_2 = \sum_{i=1}^{p} \beta_i^2$
- L_1-norm (lasso regression) : $\parallel \beta \parallel_1 = \sum_{i=1}^{p} \mid \beta_i \mid$

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

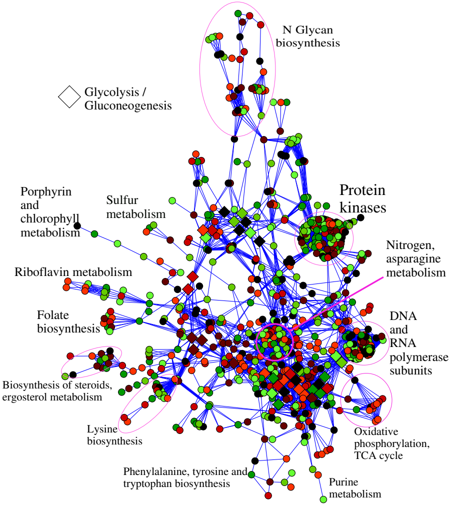
Motivation

- 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 β
- Relevant if the true decision function is indeed smooth w.r.t. the biological network

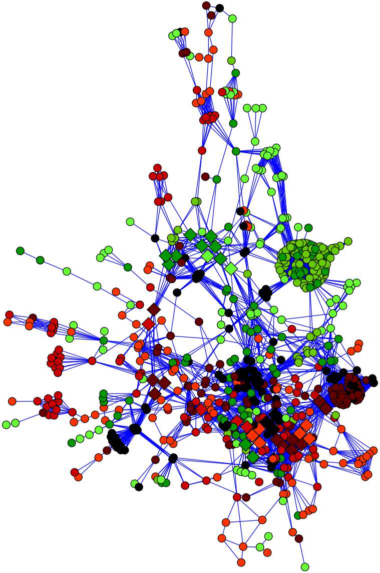
Pathway interpretation



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!

Part 6

Open Problems and Research Topics

Outline

1. Kernels and RKHS

2. Kernel tricks

3. Kernel Methods: Supervised Learning

4. Kernel Methods: Unsupervised 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

Motivation

- 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 (eg, linear, polynomial, Gaussian kernels with different bandwidths...)
  - because we have different views of the same data, eg, 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 $f^n = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)) \in \mathbb{R}^n$
- Given a p.d. kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, we learn with K by solving:

$$\min_{f \in \mathcal{H}} R(f^n) + \lambda \| f \|_{\mathcal{H}}^2 , \tag{4}$$

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 $K_1, \dots, K_M$ be M kernels on $\mathcal{X}$. The sum kernel K_S is the kernel on $\mathcal{X}$ defined as

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}\,, \quad K_S(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{M} K_i(\mathbf{x}, \mathbf{x}')\,.$$

Sum kernel and vector concatenation

Theorem

For $i = 1, \ldots, M$, let $\Phi_i : \mathcal{X} \to \mathcal{H}_i$ be a feature map such that

$$K_i(\mathbf{x}, \mathbf{x}') = \left\langle \Phi_i\left(\mathbf{x}\right), \Phi_i\left(\mathbf{x}'\right) \right\rangle_{\mathcal{H}_i} .$$

Then $K_S = \sum_{i=1}^{M} K_i$ can be written as:

$$K_S(\mathbf{x}, \mathbf{x}') = \left\langle \Phi_S\left(\mathbf{x}\right), \Phi_S\left(\mathbf{x}'\right) \right\rangle_{\mathcal{H}_S} ,$$

where $\Phi_S : \mathcal{X} \to \mathcal{H}_S = \mathcal{H}_1 \oplus \ldots \oplus \mathcal{H}_M$ is the concatenation of the feature maps Φ_i:

$$\Phi_S\left(\mathbf{x}\right) = \left(\Phi_1\left(\mathbf{x}\right), \ldots, \Phi_M\left(\mathbf{x}\right)\right)^\top .$$

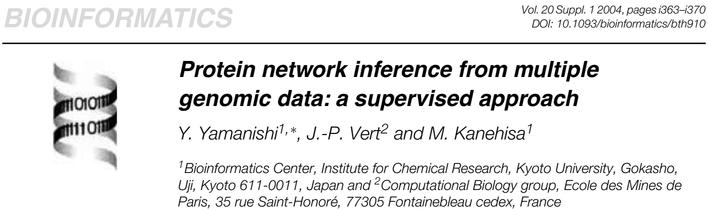
Therefore, summing kernels amounts to concatenating their feature space representations, which is a quite natural way to integrate different features.

Proof

For $\Phi_S(\mathbf{x}) = (\Phi_1(\mathbf{x}), \ldots, \Phi_M(\mathbf{x}))^\top$, we easily compute:

$$\begin{aligned}
\left\langle \Phi_S(\mathbf{x}), \Phi_S(\mathbf{x}') \right\rangle_{\mathcal{H}_s} &= \sum_{i=1}^{M} \left\langle \Phi_i(\mathbf{x}), \Phi_i(\mathbf{x}') \right\rangle_{\mathcal{H}_i} \\
&= \sum_{i=1}^{M} K_i(\mathbf{x}, \mathbf{x}') \\
&= K_S(\mathbf{x}, \mathbf{x}') \,.
\end{aligned}$$

Example: data integration with the sum kernel



BIOINFORMATICS

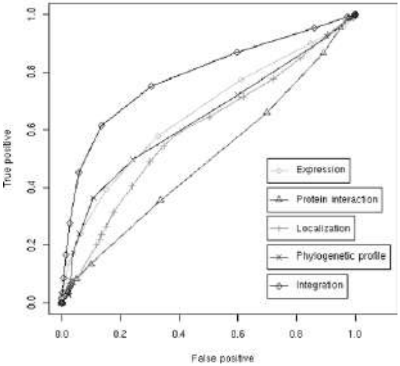
Vol. 20 Suppl. 1 2004, pages i363–i370
DOI: 10.1093/bioinformatics/bth910

Protein network inference from multiple genomic data: a supervised approach

Y. Yamanishi[1],*, J.-P. Vert[2] and M. Kanehisa[1]

[1]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611-0011, Japan and [2]Computational Biology group, Ecole des Mines de Paris, 35 rue Saint-Honoré, 77305 Fontainebleau cedex, France

K_{exp} (Expression)
K_{ppi} (Protein interaction)
K_{loc} (Localization)
K_{phy} (Phylogenetic profile)
$K_{\text{exp}} + K_{\text{ppi}} + K_{\text{loc}} + K_{\text{phy}}$
(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:

$$\min_{f_1,\ldots,f_M} R\left(\sum_{i=1}^{M} f_i^n\right) + \lambda \sum_{i=1}^{M} \| f_i \|_{\mathcal{H}_{K_i}}^2 \,.$$

Generalization: The weighted sum kernel

Theorem

The solution f^* when we learn with $K_{\boldsymbol{\eta}} = \sum_{i=1}^{M} \eta_i K_i$, 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)

$$\min_{f_1,\ldots,f_M} R\left(\sum_{i=1}^{M} f_i^n\right) + \lambda \sum_{i=1}^{M} \frac{\| f_i \|_{\mathcal{H}_{K_i}}^2}{\eta_i} .$$

- 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 $\boldsymbol{\alpha}_1^*, \ldots, \boldsymbol{\alpha}_M^* \in \mathbb{R}^n$ such that

$$f_i^*(\mathbf{x}) = \sum_{j=1}^{n} \alpha_{ij}^* K_i(\mathbf{x}_j, \mathbf{x}) .$$

- $(\boldsymbol{\alpha}_1^*, \ldots, \boldsymbol{\alpha}_M^*)$ is the solution of

$$\min_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_M \in \mathbb{R}^n} R\left(\sum_{i=1}^{M} \mathbf{K}_i \boldsymbol{\alpha}_i\right) + \lambda \sum_{i=1}^{M} \frac{\boldsymbol{\alpha}_i^\top \mathbf{K}_i \boldsymbol{\alpha}_i}{\eta_i} .$$

Proof (2/4)

- This is equivalent to

$$\min_{\mathbf{u},\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_M\in\mathbb{R}^n} R\left(\mathbf{u}\right)+\lambda\sum_{i=1}^{M}\frac{\boldsymbol{\alpha}_i^\top\mathbf{K}_i\boldsymbol{\alpha}_i}{\eta_i} \quad \text{s.t.} \quad u=\sum_{i=1}^{M}\mathbf{K}_i\boldsymbol{\alpha}_i\,.$$

- This is equivalent to the saddle point problem:

$$\min_{\mathbf{u},\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_M\in\mathbb{R}^n}\max_{\boldsymbol{\gamma}\in\mathbb{R}^n} R\left(\mathbf{u}\right)+\lambda\sum_{i=1}^{M}\frac{\boldsymbol{\alpha}_i^\top\mathbf{K}_i\boldsymbol{\alpha}_i}{\eta_i}+2\lambda\boldsymbol{\gamma}^\top(\mathbf{u}-\sum_{i=1}^{M}\mathbf{K}_i\boldsymbol{\alpha}_i)\,.$$

- By Slater's condition, strong duality holds, meaning we can invert min and max:

$$\max_{\boldsymbol{\gamma}\in\mathbb{R}^n}\min_{\mathbf{u},\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_M\in\mathbb{R}^n} R\left(\mathbf{u}\right)+\lambda\sum_{i=1}^{M}\frac{\boldsymbol{\alpha}_i^\top\mathbf{K}_i\boldsymbol{\alpha}_i}{\eta_i}+2\lambda\boldsymbol{\gamma}^\top(\mathbf{u}-\sum_{i=1}^{M}\mathbf{K}_i\boldsymbol{\alpha}_i)\,.$$

Proof (3/4)

- Minimization in $\mathbf{u}$:

$$\min_{u} R(\mathbf{u}) + 2\lambda\boldsymbol{\gamma}^\top \mathbf{u} = -\max_{\mathbf{u}} \left\{ -2\lambda\boldsymbol{\gamma}^\top \mathbf{u} - R(\mathbf{u}) \right\} = -R^*(-2\lambda\boldsymbol{\gamma}),$$

where R^* is the Fenchel dual of R:

$$\forall \mathbf{v} \in \mathbb{R}^n \quad R^*(\mathbf{v}) = \sup_{\mathbf{u}\in\mathbb{R}^n} \mathbf{u}^\top \mathbf{v} - R(\mathbf{u}).$$

- Minimization in $\boldsymbol{\alpha}_i$ for $i = 1, \ldots, M$:

$$\min_{\boldsymbol{\alpha}_i} \left\{ \lambda \frac{\boldsymbol{\alpha}_i^\top \mathbf{K}_i \boldsymbol{\alpha}_i}{\eta_i} - 2\lambda\boldsymbol{\gamma}^\top \mathbf{K}_i \boldsymbol{\alpha}_i \right\} = -\lambda\eta_i\boldsymbol{\gamma}^\top \mathbf{K}_i \boldsymbol{\gamma},$$

where the minimum in $\boldsymbol{\alpha}_i$ is reached for $\boldsymbol{\alpha}_i^* = \eta_i\boldsymbol{\gamma}$.

Proof (4/4)

- The dual problem is therefore

$$\max_{\boldsymbol{\gamma}\in\mathbb{R}^n}\left\{-R^*(-2\lambda\boldsymbol{\gamma})-\lambda\boldsymbol{\gamma}^\top\left(\sum_{i=1}^M \eta_i\mathbf{K}_i\right)\boldsymbol{\gamma}\right\}.$$

- Note that if learn from a single kernel $\mathbf{K}_{\boldsymbol{\eta}}$, we get the same dual problem

$$\max_{\boldsymbol{\gamma}\in\mathbb{R}^n}\left\{-R^*(-2\lambda\boldsymbol{\gamma})-\lambda\boldsymbol{\gamma}^\top\mathbf{K}_{\boldsymbol{\eta}}\boldsymbol{\gamma}\right\}.$$

- If $\boldsymbol{\gamma}^*$ is a solution of the dual problem, then $\boldsymbol{\alpha}_i^*=\eta_i\boldsymbol{\gamma}^*$ leading to:

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

- Therefore, $f^*=\sum_{i=1}^M f_i^*$ satisfies

$$f^*(\mathbf{x})=\sum_{i=1}^M\sum_{j=1}^n\eta_i\gamma_j^*\mathbf{K}_i(\mathbf{x}_j,\mathbf{x})=\sum_{j=1}^n\gamma_j^*\mathbf{K}_\eta(\mathbf{x}_j,\mathbf{x})\,.\quad\square$$

Learning the kernel

Motivation

- If we know how to weight each kernel, then we can learn with the weighted kernel

$$\mathbf{K}_{\boldsymbol{\eta}} = \sum_{i=1}^{M} \eta_i \mathbf{K}_i$$

- However, usually we don't know...
- Perhaps we can optimize the weights η_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

$$J(K) = \max_{\boldsymbol{\gamma} \in \mathbb{R}^n} \left\{ -R^*(-2\lambda\boldsymbol{\gamma}) - \lambda\gamma^\top \mathbf{K}\boldsymbol{\gamma} \right\} .$$

- For each γ 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

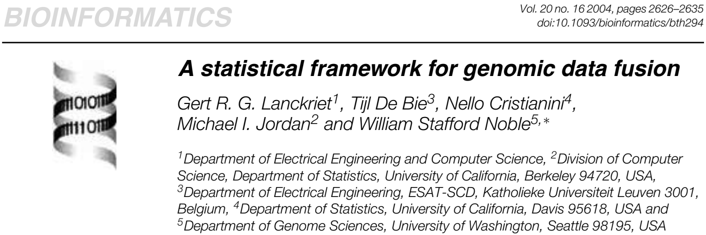
$$K_{\boldsymbol{\eta}} = \sum_{i=1}^{M} \eta_i K_i \quad \text{with} \quad \boldsymbol{\eta} \in \Sigma_M = \left\{ \eta_i \geq 0\,, \sum_{i=1}^{M} \eta_i = 1 \right\}$$

- We optimize both $\boldsymbol{\eta}$ and f^* by solving:

$$\min_{\boldsymbol{\eta} \in \Sigma_M} J\left(K_{\boldsymbol{\eta}}\right) = \min_{\boldsymbol{\eta} \in \Sigma_M} \min_{f \in \mathcal{H}_{K_{\boldsymbol{\eta}}}} \left\{ R(f^n) + \lambda \| f \|^2_{\mathcal{H}_{K_{\boldsymbol{\eta}}}} \right\}$$

- The problem is jointly convex in $(\boldsymbol{\eta}, \boldsymbol{\alpha})$ and can be solved efficiently.
- The output is both a set of weights $\boldsymbol{\eta}$, and a predictor corresponding to the kernel method trained with kernel $K_{\boldsymbol{\eta}}$.
- This method is usually called Multiple Kernel Learning (MKL).

Example: protein annotation



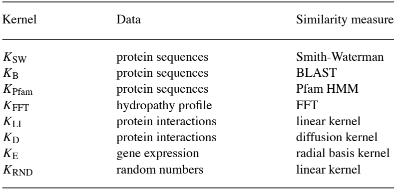
BIOINFORMATICS

Vol. 20 no. 16 2004, pages 2626–2635
doi:10.1093/bioinformatics/bth294

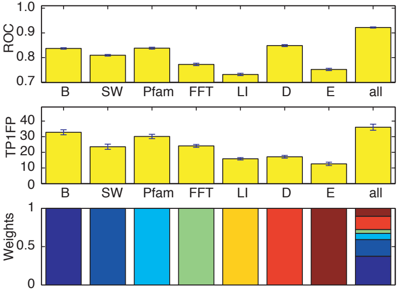
A statistical framework for genomic data fusion

Gert R. G. Lanckriet[1], Tijl De Bie[3], Nello Cristianini[4],
*Michael I. Jordan[2] and William Stafford Noble[5,]**

[1]Department of Electrical Engineering and Computer Science, [2]Division of Computer Science, Department of Statistics, University of California, Berkeley 94720, USA, [3]Department of Electrical Engineering, ESAT-SCD, Katholieke Universiteit Leuven 3001, Belgium, [4]Department of Statistics, University of California, Davis 95618, USA and [5]Department of Genome Sciences, University of Washington, Seattle 98195, USA



Kernel	Data	Similarity measure
K_{SW}	protein sequences	Smith-Waterman
K_{B}	protein sequences	BLAST
K_{Pfam}	protein sequences	Pfam HMM
K_{FFT}	hydropathy profile	FFT
K_{LI}	protein interactions	linear kernel
K_{D}	protein interactions	diffusion kernel
K_{E}	gene expression	radial basis kernel
K_{RND}	random numbers	linear kernel

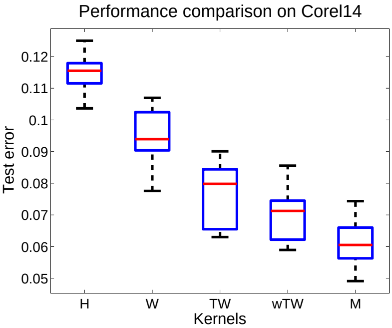


(B) Membrane proteins

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).



MKL revisited (Bach et al., 2004)

$$K_{\boldsymbol{\eta}} = \sum_{i=1}^{M} \eta_i K_i \quad \text{with} \quad \boldsymbol{\eta} \in \Sigma_M = \left\{ \eta_i \geq 0 \, , \sum_{i=1}^{M} \eta_i = 1 \right\}$$

Theorem

The solution f^* of

$$\min_{\boldsymbol{\eta} \in \Sigma_M} \min_{f \in \mathcal{H}_{K_{\boldsymbol{\eta}}}} \left\{ R(f^n) + \lambda \| \, f \, \|^2_{\mathcal{H}_{K_{\boldsymbol{\eta}}}} \right\}$$

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)

$$
\begin{aligned}
\min_{\boldsymbol{\eta}\in\boldsymbol{\Sigma}_M} \min_{f\in\mathcal{H}_{K_{\boldsymbol{\eta}}}} & \left\{ R(f^n) + \lambda \| f \|^2_{\mathcal{H}_{K_{\boldsymbol{\eta}}}} \right\} \\
&= \min_{\boldsymbol{\eta}\in\boldsymbol{\Sigma}_M} \min_{f_1,\ldots,f_M} \left\{ R\left(\sum_{i=1}^{M} f_i^n \right) + \lambda \sum_{i=1}^{M} \frac{\| f_i \|^2_{\mathcal{H}_{K_i}}}{\eta_i} \right\} \\
&= \min_{f_1,\ldots,f_M} \left\{ R\left(\sum_{i=1}^{M} f_i^n \right) + \lambda \min_{\boldsymbol{\eta}\in\boldsymbol{\Sigma}_M} \left\{ \sum_{i=1}^{M} \frac{\| f_i \|^2_{\mathcal{H}_{K_i}}}{\eta_i} \right\} \right\} \\
&= \min_{f_1,\ldots,f_M} \left\{ R\left(\sum_{i=1}^{M} f_i^n \right) + \lambda \left(\sum_{i=1}^{M} \| f_i \|_{\mathcal{H}_{K_i}} \right)^2 \right\} ,
\end{aligned}
$$

Proof (2/2)

where the last equality results from:

$$\forall \mathbf{a} \in \mathbb{R}_+^M \,, \quad \left(\sum_{i=1}^M a_i \right)^2 = \inf_{\boldsymbol{\eta} \in \Sigma_M} \sum_{i=1}^M \frac{a_i^2}{\eta_i} \,,$$

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 $\boldsymbol{\eta} \in \Sigma_M$:

$$\min_{\boldsymbol{\eta}\in\Sigma_M} J\left(K_{\boldsymbol{\eta}}\right) = \min_{\boldsymbol{\eta}\in\Sigma_M} \max_{\boldsymbol{\gamma}\in\mathbb{R}^n} \left\{ -R^*(-2\lambda\boldsymbol{\gamma}) - \lambda\boldsymbol{\gamma}^\top \mathbf{K}_{\boldsymbol{\eta}}\boldsymbol{\gamma} \right\} .$$

- For a fixed $\boldsymbol{\eta} \in \Sigma_M$, we can compute $f(\boldsymbol{\eta}) = J\left(K_{\boldsymbol{\eta}}\right)$ by using a standard solver for a single kernel to find γ^*:

$$J\left(K_{\boldsymbol{\eta}}\right) = -R^*(-2\lambda\boldsymbol{\gamma}^*) - \lambda\boldsymbol{\gamma}^{*\top} \mathbf{K}_{\boldsymbol{\eta}}\boldsymbol{\gamma}^* .$$

- From $\boldsymbol{\gamma}^*$ we can also compute the gradient of $J\left(K_{\boldsymbol{\eta}}\right)$ with respect to $\boldsymbol{\eta}$:

$$\frac{\partial J\left(K_{\boldsymbol{\eta}}\right)}{\partial \eta_i} = -\lambda\boldsymbol{\gamma}^{*\top} K_i \boldsymbol{\gamma}^* .$$

- $J\left(K_{\boldsymbol{\eta}}\right)$ can then be minimized on Σ_M by a projected gradient or reduced gradient algorithm.

Sum kernel vs MKL

- Learning with the sum kernel (uniform combination) solves

$$\min_{f_1,\dots,f_M} \left\{ R\left(\sum_{i=1}^M f_i^n\right) + \lambda \sum_{i=1}^M \| f_i \|_{\mathcal{H}_{K_i}}^2 \right\} .$$

- Learning with MKL (best convex combination) solves

$$\min_{f_1,\dots,f_M} \left\{ R\left(\sum_{i=1}^M f_i^n\right) + \lambda \left(\sum_{i=1}^M \| f_i \|_{\mathcal{H}_{K_i}}\right)^2 \right\} .$$

- 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 group lasso penalty:

$$\Omega(f) = \min_{f_1+\dots+f_M=f} \sum_{i=1}^M \| f_i \|_{\mathcal{H}_{K_i}} .$$

Example: ridge vs LASSO regression

- Take $\mathcal{X} = \mathbb{R}^d$, and for $\mathbf{x} = (x_1, \ldots, x_d)^\top$ consider the rank-1 kernels:

$$\forall i = 1, \ldots, d \,, \quad K_i\left(\mathbf{x}, \mathbf{x}'\right) = x_i x_i' \,.$$

- A function $f_i \in \mathcal{H}_{K_i}$ has the form $f_i(\mathbf{x}) = \beta_i x_i$, with $\parallel f_i \parallel_{\mathcal{H}_{K_i}} = \mid \beta_i \mid$
- The sum kernel is $K_S(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^d x_i x_i' = \mathbf{x}^\top \mathbf{x}$, a function $\mathcal{H}_{K_S}$ is of the form $f(\mathbf{x}) = \boldsymbol{\beta}^\top \mathbf{x}$, with norm $\parallel f \parallel_{\mathcal{H}_{K_S}} = \parallel \boldsymbol{\beta} \parallel_{\mathbb{R}^d}$.
- Learning with the sum kernel solves a ridge regression problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \left\{ R(\mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{i=1}^d \beta_i^2 \right\} .$$

- Learning with MKL solves a LASSO regression problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \left\{ R(\mathbf{X}\boldsymbol{\beta}) + \lambda \left(\sum_{i=1}^d \mid \beta_i \mid \right)^2 \right\} .$$

Extensions (Micchelli et al., 2005)

$$\text{For } r > 0\,, \quad K_{\boldsymbol{\eta}} = \sum_{i=1}^{M} \eta_i K_i \quad \text{with} \quad \boldsymbol{\eta} \in \Sigma_M^r = \left\{ \eta_i \geq 0\,, \sum_{i=1}^{M} \eta_i^r = 1 \right\}$$

Theorem

The solution f^* of

$$\min_{\boldsymbol{\eta} \in \Sigma_M^r} \min_{f \in \mathcal{H}_{K_{\boldsymbol{\eta}}}} \left\{ R(f^n) + \lambda \| f \|^2_{\mathcal{H}_{K_{\boldsymbol{\eta}}}} \right\}$$

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:

$$\min_{f_1, \ldots, f_M} \left\{ R\left(\sum_{i=1}^{M} f_i^n \right) + \lambda \left(\sum_{i=1}^{M} \| f_i \|_{\mathcal{H}_{K_i}}^{\frac{2r}{r+1}} \right)^{\frac{r+1}{r}} \right\} .$$

Outline

1. Kernels and RKHS
2. Kernel tricks
3. Kernel Methods: Supervised Learning
4. Kernel Methods: Unsupervised 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

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

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

$$K(\mathbf{x}, \mathbf{x}') \approx \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle_{\mathbb{R}^d}.$$

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.,.

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \approx \left\langle \sum_{i=1}^{n} \alpha_i \psi(\mathbf{x}_i), \psi(\mathbf{x}) \right\rangle_{\mathbb{R}^d} = \langle \mathbf{w}, \psi(\mathbf{x}) \rangle_{\mathbb{R}^d}.$$

Then, the ERM problem

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

becomes, approximately,

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \mathbf{w}^\top \psi(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|_2^2,$$

which we know how to solve when n is large.

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

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.

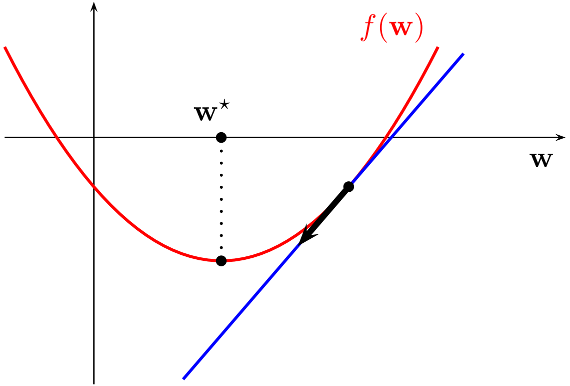
Introduction of a few optimization principles

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

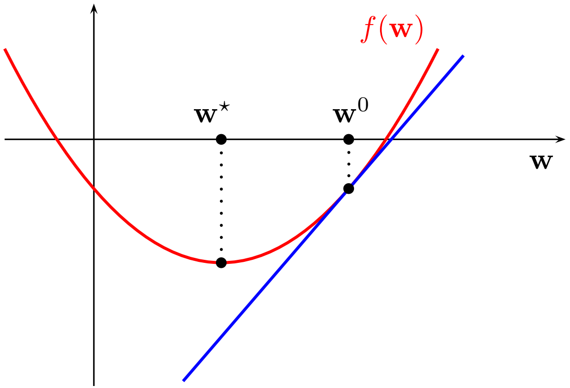


- $\nabla f(\mathbf{w}) = 0$ is a necessary and sufficient optimality condition for differentiable convex functions;
- it is often easy to upper-bound $f(\mathbf{w}) - f^\star$.

Introduction of a few optimization principles

An important inequality for smooth convex functions

If f is convex

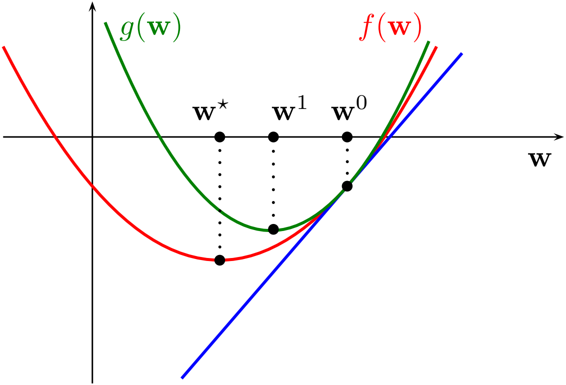


- $f(\mathbf{w}) \geq \underbrace{f(\mathbf{w}^0) + \nabla f(\mathbf{w}^0)^\top(\mathbf{w} - \mathbf{w}^0)}_{\text{linear approximation}}$;
- this is an equivalent definition of convexity for smooth functions.

Introduction of a few optimization principles

An important inequality for smooth functions

If ∇f is L-Lipschitz continuous (f does not need to be convex)

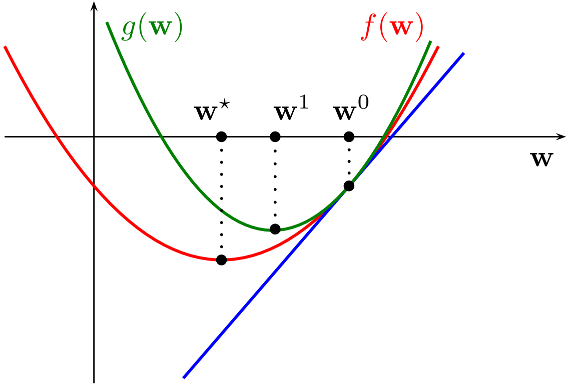


- $f(\mathbf{w}) \leq g(\mathbf{w}) = f(\mathbf{w}^0) + \nabla f(\mathbf{w}^0)^\top(\mathbf{w} - \mathbf{w}^0) + \frac{L}{2}\|\mathbf{w} - \mathbf{w}^0\|_2^2$;
- $g(\mathbf{w}) = C_{\mathbf{w}^0} + \frac{L}{2}\|\mathbf{w}^0 - (1/L)\nabla f(\mathbf{w}^0) - \mathbf{w}\|_2^2$.

Introduction of a few optimization principles

An important inequality for smooth functions

If ∇f is L-Lipschitz continuous (f does not need to be convex)



- $f(\mathbf{w}) \leq g(\mathbf{w}) = f(\mathbf{w}^0) + \nabla f(\mathbf{w}^0)^\top(\mathbf{w} - \mathbf{w}^0) + \frac{L}{2}\|\mathbf{w} - \mathbf{w}^0\|_2^2$;
- $\boxed{\mathbf{w}^1 = \mathbf{w}^0 - \frac{1}{L}\nabla f(\mathbf{w}^0) \text{ (gradient descent step).}}$

Introduction of a few optimization principles

Gradient Descent Algorithm

Assume that f is convex and differentiable, and that ∇f is L-Lipschitz.

Theorem

Consider the algorithm

$$\boxed{\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \tfrac{1}{L}\nabla f(\mathbf{w}^{t-1}).}$$

Then,

$$f(\mathbf{w}^t) - f^\star \leq \frac{L\|\mathbf{w}^0 - \mathbf{w}^\star\|_2^2}{2t}.$$

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**,

$$f(\mathbf{w}) \leq f(\mathbf{z}) + \nabla f(\mathbf{z})^\top (\mathbf{w} - \mathbf{z}) + \frac{L}{2} \|\mathbf{w} - \mathbf{z}\|_2^2.$$

Proof (1/2)

Proof of the main inequality for smooth functions

We want to show that for all **w** and **z**,

$$f(\mathbf{w}) \leq f(\mathbf{z}) + \nabla f(\mathbf{z})^\top(\mathbf{w} - \mathbf{z}) + \frac{L}{2}\|\mathbf{w} - \mathbf{z}\|_2^2.$$

By using Taylor's theorem with integral form,

$$f(\mathbf{w}) - f(\mathbf{z}) = \int_0^1 \nabla f(t\mathbf{w} + (1 - t)\mathbf{z})^\top(\mathbf{w} - \mathbf{z})dt.$$

Then,

$$\begin{aligned} f(\mathbf{w})-f(\mathbf{z})-\nabla f(\mathbf{z})^\top(\mathbf{w}-\mathbf{z}) &\leq \int_0^1 (\nabla f(t\mathbf{w}+(1-t)\mathbf{z})-\nabla f(\mathbf{z}))^\top(\mathbf{w}-\mathbf{z})dt \\ &\leq \int_0^1 |(\nabla f(t\mathbf{w}+(1-t)\mathbf{z})-\nabla f(\mathbf{z}))^\top(\mathbf{w}-\mathbf{z})|dt \\ &\leq \int_0^1 \|\nabla f(t\mathbf{w}+(1-t)\mathbf{z})-\nabla f(\mathbf{z})\|_2\|\mathbf{w}-\mathbf{z}\|_2dt \quad \text{(C.-S.)} \\ &\leq \int_0^1 Lt\|\mathbf{w}-\mathbf{z}\|_2^2dt = \frac{L}{2}\|\mathbf{w}-\mathbf{z}\|_2^2. \end{aligned}$$

Proof (2/2)

Proof of the theorem

We have shown that for all $\mathbf{w}$,

$$f(\mathbf{w}) \leq g_t(\mathbf{w}) = f(\mathbf{w}^{t-1}) + \nabla f(\mathbf{w}^{t-1})^\top(\mathbf{w} - \mathbf{w}^{t-1}) + \frac{L}{2}\|\mathbf{w} - \mathbf{w}^{t-1}\|_2^2.$$

g_t is minimized by $\mathbf{w}^t$; it can be rewritten $g_t(\mathbf{w}) = g_t(\mathbf{w}^t) + \frac{L}{2}\|\mathbf{w} - \mathbf{w}^t\|_2^2$. Then,

$$\begin{aligned} f(\mathbf{w}^t) &\leq g_t(\mathbf{w}^t) = g_t(\mathbf{w}^\star) - \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^t\|_2^2 \\ &= f(\mathbf{w}^{t-1}) + \nabla f(\mathbf{w}^{t-1})^\top(\mathbf{w}^\star - \mathbf{w}^{t-1}) + \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^{t-1}\|_2^2 - \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^t\|_2^2 \\ &\leq f^\star + \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^{t-1}\|_2^2 - \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^t\|_2^2. \end{aligned}$$

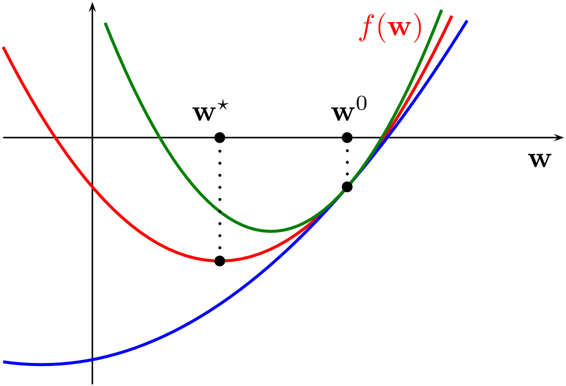
By summing from $t = 1$ to T, we have a telescopic sum

$$T(f(\mathbf{w}^T) - f^\star) \leq \sum_{t=1}^{T} f(\mathbf{w}^t) - f^\star \leq \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^0\|_2^2 - \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^T\|_2^2.$$

Introduction of a few optimization principles

An important inequality for smooth and μ-strongly convex functions

If ∇f is L-Lipschitz continuous and f μ-strongly convex



- $f(\mathbf{w}) \leq f(\mathbf{w}^0) + \nabla f(\mathbf{w}^0)^\top(\mathbf{w} - \mathbf{w}^0) + \frac{L}{2}\|\mathbf{w} - \mathbf{w}^0\|_2^2$;
- $f(\mathbf{w}) \geq f(\mathbf{w}^0) + \nabla f(\mathbf{w}^0)^\top(\mathbf{w} - \mathbf{w}^0) + \frac{\mu}{2}\|\mathbf{w} - \mathbf{w}^0\|_2^2$;

Introduction of a few optimization principles

Proposition

When f is μ-strongly convex, differentiable and ∇f is L-Lipschitz, the gradient descent algorithm with step-size $1/L$ produces iterates such that

$$f(\mathbf{w}^t) - f^\star \leq \left(1 - \frac{\mu}{L}\right)^t \frac{L\|\mathbf{w}^0 - \mathbf{w}^\star\|_2^2}{2}.$$

We call that a linear convergence rate.

Proof

We start from an inequality from the previous proof

$$\begin{aligned} f(\mathbf{w}^t) &\leq f(\mathbf{w}^{t-1}) + \nabla f(\mathbf{w}^{t-1})^\top(\mathbf{w}^\star - \mathbf{w}^{t-1}) + \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^{t-1}\|_2^2 - \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^t\|_2^2 \\ &\leq f^\star + \frac{L-\mu}{2}\|\mathbf{w}^\star - \mathbf{w}^{t-1}\|_2^2 - \frac{L}{2}\|\mathbf{w}^\star - \mathbf{w}^t\|_2^2. \end{aligned}$$

In addition, we have that $f(\mathbf{w}^t) \geq f^\star + \frac{\mu}{2}\|\mathbf{w}^t - \mathbf{w}^\star\|_2^2$, and thus

$$\begin{aligned} \|\mathbf{w}^\star - \mathbf{w}^t\|_2^2 &\leq \frac{L-\mu}{L+\mu}\|\mathbf{w}^\star - \mathbf{w}^{t-1}\|_2^2 \\ &\leq \left(1 - \frac{\mu}{L}\right.^{\boldsymbol{\eta}} \|\mathbf{w}^\star - \mathbf{w}^{t-1}\|_2^2. \end{aligned}$$

Finally,

$$\begin{aligned} f(\mathbf{w}^t) - f^\star &\leq \frac{L}{2}\|\mathbf{w}^t - \mathbf{w}^\star\|_2^2 \\ &\leq \left(1 - \frac{\mu}{L}\right.^{\boldsymbol{\eta}_t} \frac{L\|\mathbf{w}^\star - \mathbf{w}^0\|_2^2}{2} \end{aligned}$$

The stochastic (sub)gradient descent algorithm

Consider now the minimization of an expectation

$$\min_{\mathbf{w}\in\mathbb{R}^p} f(\mathbf{w}) = \mathbb{E}_{\mathbf{x}}[\ell(\mathbf{x},\mathbf{w})],$$

To simplify, we assume that for all $\mathbf{x}$, $\mathbf{w} \mapsto \ell(\mathbf{x},\mathbf{w})$ is differentiable, but everything here is true for nonsmooth functions.

Algorithm

At iteration t,

- Randomly draw one example $\mathbf{x}_t$ from the training set;
- Update the current iterate

$$\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \eta_t \nabla_{\mathbf{w}} \ell(\mathbf{x}_t, \mathbf{w}_{t-1}).$$

- Perform online averaging of the iterates (optional)

$$\tilde{\mathbf{w}}^t \leftarrow (1-\gamma_t)\tilde{\mathbf{w}}^{t-1} + \gamma_t \mathbf{w}^t.$$

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 η_t, γ_t, classical convergence rates may be obtained:

- $f(\tilde{\mathbf{w}}^t) - f^\star = O(1/\sqrt{t})$ for convex problems;
- $f(\tilde{\mathbf{w}}^t) - f^\star = O(1/t)$ 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:

$$\min_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{w}),$$

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

$$\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \frac{\gamma}{Ln} \sum_{i=1}^{n} \mathbf{y}_i^t \quad \text{with} \quad \mathbf{y}_i^t = \begin{cases} \nabla f_i(\mathbf{w}^{t-1}) & \text{if } \; i = i_t \\ \mathbf{y}_i^{t-1} & \text{otherwise} \end{cases} .$$

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

$$f(\mathbf{w}_t) - f^\star = O\left(\left(1 - C \max\left(\frac{1}{n}, \frac{\mu}{L}\right)\right)^t\right)$$

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).

Large-scale learning with linear models

Conclusion

- we know how to deal with huge-scale linear problems;
- this is also useful to learn with kernels!

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

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

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

The Nyström method consists of replacing any point $\varphi(\mathbf{x})$ in $\mathcal{H}$, for $\mathbf{x}$ in $\mathcal{X}$ by its orthogonal projection onto a finite-dimensional subspace

$$\mathcal{F} := \text{Span}\left(f_1, \ldots, f_p\right) \quad \text{with } p \ll n,$$

where the f_i's are anchor points in $\mathcal{H}$ (to be defined later).

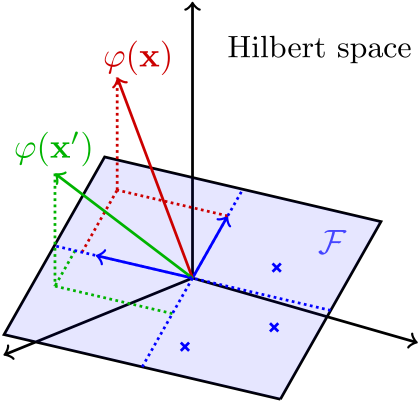
Motivation

- 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

$$\Pi_{\mathcal{F}}[\mathbf{x}] := \operatorname*{argmin}_{f \in \mathcal{F}} \|\varphi(\mathbf{x}) - f\|_{\mathcal{H}}^2 ,$$



Nyström approximations: principle

The projection is equivalent to

$$\Pi_{\mathcal{F}}[\mathbf{x}] := \sum_{j=1}^{p} \beta_j^\star f_j \quad \text{with} \quad \boldsymbol{\beta}^\star \in \operatorname*{argmin}_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\| \varphi(\mathbf{x}) - \sum_{j=1}^{p} \beta_j f_j \right\|_{\mathcal{H}}^2,$$

and $\boldsymbol{\beta}^\star$ is the solution of the problem

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} -2 \sum_{j=1}^{p} \beta_j \langle f_j, \varphi(\mathbf{x}) \rangle_{\mathcal{H}} + \sum_{j,l=1}^{p} \beta_j \beta_l \langle f_j, f_l \rangle_{\mathcal{H}},$$

or also

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} -2 \sum_{j=1}^{p} \beta_j f_j(\mathbf{x}) + \sum_{j,l=1}^{p} \beta_j \beta_l \langle f_j, f_l \rangle_{\mathcal{H}}.$$

Nyström approximations: principle

Then, call $[\mathbf{K_f}]_{jl} = \langle f_j, f_l \rangle_{\mathcal{H}}$ and $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_p(\mathbf{x})]$ in $\mathbb{R}^p$. The problem may be rewritten as

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} -2\boldsymbol{\beta}^\top \mathbf{f}(\mathbf{x}) + \boldsymbol{\beta}^\top \mathbf{K_f} \boldsymbol{\beta},$$

and, assuming $\mathbf{K_f}$ to be non-singular for simplicity, the solution is $\boldsymbol{\beta}^\star(\mathbf{x}) = \mathbf{K_f}^{-1}\mathbf{f}(\mathbf{x})$. Then,

$$\varphi(\mathbf{x}) \approx \sum_{j=1}^{p} \beta_j^\star(\mathbf{x}) f_j,$$

and

$$\begin{aligned}\langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}} &\approx \left\langle \sum_{j=1}^{p} \beta_j^\star(\mathbf{x}) f_j, \sum_{j=1}^{p} \beta_j^\star(\mathbf{x}') f_j \right\rangle_{\mathcal{H}} \\ &= \sum_{j,l=1}^{p} \beta_j^\star(\mathbf{x}) \beta_l^\star(\mathbf{x}') \langle f_j, f_l \rangle_{\mathcal{H}} = \boldsymbol{\beta}^\star(\mathbf{x})^\top \mathbf{K_f} \boldsymbol{\beta}^\star(\mathbf{x}').\end{aligned}$$

Nyström approximations: principle

This allows us to define the mapping

$$\psi(\mathbf{x}) = \mathbf{K}_{\mathbf{f}}^{1/2}\boldsymbol{\beta}^{\star}(\mathbf{x}) = \mathbf{K}_{\mathbf{f}}^{-1/2}\mathbf{f}(\mathbf{x}),$$

and we have the approximation $K(\mathbf{x}, \mathbf{x}') \approx \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle_{\mathbb{R}^p}$.

Remarks

- the mapping provides low-rank approximations of the kernel matrix. Given an $n \times n$ Gram matrix $\mathbf{K}$ computed on a training set $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we have

  $$\mathbf{K} \approx \psi(\mathcal{S})^\top \psi(\mathcal{S}),$$

  where $\psi(\mathcal{S}) := [\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_n)]$.
- 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 $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathcal{X}$:

$$\min_{\substack{f_1,\ldots,f_p \in \mathcal{H} \\ \beta_{ij} \in \mathbb{R}}} \sum_{i=1}^{n} \left\| \varphi(\mathbf{x}_i) - \sum_{j=1}^{p} \beta_{ij} f_j \right\|_{\mathcal{H}}^2 .$$

Using similar calculation as before, the objective is equivalent to

$$\min_{\substack{f_1,\ldots,f_p \in \mathcal{H} \\ \boldsymbol{\beta}_i \in \mathbb{R}^p}} \sum_{i=1}^{n} -2\boldsymbol{\beta}_i^\top \mathbf{f}(\mathbf{x}_i) + \boldsymbol{\beta}_i^\top \mathbf{K_f} \boldsymbol{\beta}_i,$$

and, by minimizing with respect to all $\boldsymbol{\beta}_i$ with $\mathbf{f}$ fixed, we have that $\boldsymbol{\beta}_i = \mathbf{K_f}^{-1}\mathbf{f}(\mathbf{x}_i)$ (assuming $\mathbf{K_f}$ to be invertible), which leads to

$$\max_{f_1,\ldots,f_p \in \mathcal{H}} \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}_i)^\top \mathbf{K_f}^{-1} \mathbf{f}(\mathbf{x}_i).$$

Nyström approximation via kernel PCA

Remember the objective:

$$\max_{f_1,\ldots,f_p \in \mathcal{H}} \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}_i)^\top \mathbf{K}_{\mathbf{f}}^{-1} \mathbf{f}(\mathbf{x}_i).$$

Consider an optimal solution $\mathbf{f}^\star$ and compute the eigenvalue decomposition of $\mathbf{K}_{\mathbf{f}^\star} = \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top$. Then, define the functions

$$\mathbf{g}^\star(\mathbf{x}) := [g_1^\star(\mathbf{x}), \ldots, g_p^\star(\mathbf{x})] = \boldsymbol{\Delta}^{-1/2}\mathbf{U}^\top \mathbf{f}^\star(\mathbf{x}).$$

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:

$$\max_{f_1,\ldots,f_p \in \mathcal{H}} \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}_i)^\top \mathbf{K}_{\mathbf{f}}^{-1} \mathbf{f}(\mathbf{x}_i).$$

Consider an optimal solution $\mathbf{f}^\star$ and compute the eigenvalue decomposition of $\mathbf{K}_{\mathbf{f}^\star} = \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top$. Then, define the functions

$$\mathbf{g}^\star(\mathbf{x}) := [g_1^\star(\mathbf{x}), \ldots, g_p^\star(\mathbf{x})] = \boldsymbol{\Delta}^{-1/2}\mathbf{U}^\top \mathbf{f}^\star(\mathbf{x}).$$

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 $\mathbf{K}_{\mathbf{f}^\star}$ *and* $\mathbf{K}_{\mathbf{f}}$*.*

Nyström approximation via kernel PCA

Besides, by construction

$$
\begin{aligned}
[\mathbf{K}_{\mathbf{g}^\star}]_{jl} &:= \langle g_j^\star, g_l^\star \rangle_{\mathcal{H}} \\
&= \left\langle \frac{1}{\sqrt{\boldsymbol{\Delta}_{jj}}} \sum_{k=1}^{p} [\mathbf{U}]_{kj} f_k^\star, \frac{1}{\sqrt{\boldsymbol{\Delta}_{ll}}} \sum_{k=1}^{p} [\mathbf{U}]_{kl} f_k^\star \right\rangle_{\mathcal{H}} \\
&= \frac{1}{\sqrt{\boldsymbol{\Delta}_{jj}}} \frac{1}{\sqrt{\boldsymbol{\Delta}_{ll}}} \sum_{k,k'=1}^{p} [\mathbf{U}]_{kj} [\mathbf{U}]_{k'l} \, \langle f_k^\star, f_{k'}^\star \rangle_{\mathcal{H}} \\
&= \frac{1}{\sqrt{\boldsymbol{\Delta}_{jj}}} \frac{1}{\sqrt{\boldsymbol{\Delta}_{ll}}} \sum_{k,k'=1}^{p} [\mathbf{U}]_{kj} [\mathbf{U}]_{k'l} [\mathbf{K}_{f^\star}]_{kk'} \\
&= \frac{1}{\sqrt{\boldsymbol{\Delta}_{jj}}} \frac{1}{\sqrt{\boldsymbol{\Delta}_{ll}}} \mathbf{u}_j^\top \mathbf{K}_{f^\star} \mathbf{u}_l \\
&= \delta_{j=l}.
\end{aligned}
$$

Nyström approximation via kernel PCA

Then, $\mathbf{K}_{\mathbf{g}^\star} = \mathbf{I}$ and $\mathbf{g}^\star$ is also a solution of the problem

$$\max_{f_1,\ldots,f_p \in \mathcal{H}} \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}_i)^\top \mathbf{K}_{\mathbf{f}}^{-1} \mathbf{f}(\mathbf{x}_i),$$

since

$$\begin{aligned} \mathbf{f}^\star(\mathbf{x}_i)^\top \mathbf{K}_{\mathbf{f}^\star}^{-1} \mathbf{f}^\star(\mathbf{x}_i) &= \mathbf{f}^\star(\mathbf{x}_i)^\top \mathbf{U} \boldsymbol{\Delta}^{-1} \mathbf{U}^\top \mathbf{f}^\star(\mathbf{x}_i) \\ &= \mathbf{g}^\star(\mathbf{x}_i)^\top \mathbf{g}^\star(\mathbf{x}_i) = \mathbf{g}^\star(\mathbf{x}_i)^\top \mathbf{K}_{\mathbf{g}^\star}^{-1} \mathbf{g}^\star(\mathbf{x}_i), \end{aligned}$$

and also a solution of the problem

$$\max_{g_1,\ldots,g_p \in \mathcal{H}} \sum_{j=1}^{p} \sum_{i=1}^{n} g_j(\mathbf{x}_i)^2 \quad \text{s.t.} \quad g_j \perp g_k \quad \text{for} \quad k \neq j \quad \text{and} \quad \|g_j\|_{\mathcal{H}} = 1.$$

Nyström approximation via kernel PCA

Then, $\mathbf{K}_{\mathbf{g}^\star} = \mathbf{I}$ and $\mathbf{g}^\star$ is also a solution of the problem

$$\max_{f_1,\dots,f_p \in \mathcal{H}} \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}_i)^\top \mathbf{K}_{\mathbf{f}}^{-1} \mathbf{f}(\mathbf{x}_i),$$

since

$$\begin{aligned} \mathbf{f}^\star(\mathbf{x}_i)^\top \mathbf{K}_{\mathbf{f}^\star}^{-1} \mathbf{f}^\star(\mathbf{x}_i) &= \mathbf{f}^\star(\mathbf{x}_i)^\top \mathbf{U} \boldsymbol{\Delta}^{-1} \mathbf{U}^\top \mathbf{f}^\star(\mathbf{x}_i) \\ &= \mathbf{g}^\star(\mathbf{x}_i)^\top \mathbf{g}^\star(\mathbf{x}_i) = \mathbf{g}^\star(\mathbf{x}_i)^\top \mathbf{K}_{\mathbf{g}^\star}^{-1} \mathbf{g}^\star(\mathbf{x}_i), \end{aligned}$$

and also a solution of the problem

$$\max_{g_1,\dots,g_p \in \mathcal{H}} \sum_{j=1}^{p} \sum_{i=1}^{n} g_j(\mathbf{x}_i)^2 \quad \text{s.t.} \quad g_j \perp g_k \quad \text{for} \quad k \neq j \quad \text{and} \quad \|g_j\|_{\mathcal{H}} = 1.$$

This is the kernel PCA formulation!

Nyström approximation via kernel PCA

Our first recipe with kernel PCA

Given a dataset of n training points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathcal{X}$,

- randomly choose a subset $\mathcal{Z} = [\mathbf{x}_{z_1}, \ldots, \mathbf{x}_{z_m}]$ of $m \leq n$ training points;
- compute the $m \times m$ kernel matrix $\mathbf{K}_{\mathcal{Z}}$.
- perform kernel PCA to find the $p \leq m$ largest principal directions (parametrized by p vectors $\boldsymbol{\alpha}_j$ in $\mathbb{R}^m$);

Then, every point $\mathbf{x}$ in $\mathcal{X}$ may be approximated by

$$\psi(\mathbf{x}) = \mathbf{K}_{\mathbf{g}^\star}^{-1/2}\mathbf{g}^\star(\mathbf{x}) = \mathbf{g}^\star(\mathbf{x}) = [g_1^\star(\mathbf{x}), \ldots, g_p^\star(\mathbf{x})]^\top$$

$$= \left[\sum_{i=1}^{m} \alpha_{1i} K(\mathbf{x}_{z_i}, \mathbf{x}), \ldots, \sum_{i=1}^{m} \alpha_{pi} K(\mathbf{x}_{z_i}, \mathbf{x})\right]^\top.$$

Nyström approximation via kernel PCA

Remarks

- The vector $\psi(\mathbf{x})$ can be interpreted as coordinates of the projection of $\varphi(\mathbf{x})$ onto the (orthogonal) PCA basis.
- The complexity of training is $O(m^3)$ (eig decomposition of $\mathbf{K}_{\mathcal{Z}}$) + $O(m^2)$ kernel evaluations.
- The complexity of encoding a new point $\mathbf{x}$ is $O(mp)$ (matrix vector multiplication) + $O(m)$ kernel evaluations.

Nyström approximation via kernel PCA

Remarks

- The vector $\psi(\mathbf{x})$ can be interpreted as coordinates of the projection of $\varphi(\mathbf{x})$ onto the (orthogonal) PCA basis.
- The complexity of training is $O(m^3)$ (eig decomposition of $\mathbf{K}_{\mathcal{Z}}$) + $O(m^2)$ kernel evaluations.
- The complexity of encoding a new point $\mathbf{x}$ 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 $\mathbf{x}_1, \dots, \mathbf{x}_n$—that is,

$$\mathcal{F} := \mathrm{span}(\varphi(\mathbf{x}_{z_1}), \dots, \varphi(\mathbf{z}_{z_p})).$$

In other words, choose $f_1 = \varphi(\mathbf{x}_{z_1}), \dots, f_p = \varphi(\mathbf{x}_{z_p})$.

Second recipe with random point sampling

Given a dataset of n training points $\mathbf{x}_1, \dots, \mathbf{x}_n$ in $\mathcal{X}$,

- randomly choose a subset $\mathcal{Z} = [\mathbf{x}_{z_1}, \dots, \mathbf{x}_{z_p}]$ of p training points;
- compute the $p \times p$ kernel matrix $\mathbf{K}_{\mathcal{Z}}$.

Then, a new point $\mathbf{x}$ is encoded as

$$\begin{aligned}\psi(\mathbf{x}) &= \mathbf{K}_{\mathcal{Z}}^{-1/2}\mathbf{f}_{\mathcal{Z}}(\mathbf{x}) \\ &= \mathbf{K}_{\mathcal{Z}}^{-1/2}[K(\mathbf{x}_{z_1}, \mathbf{x}), \dots, K(\mathbf{x}_{z_p}, \mathbf{x})]^\top\end{aligned}$$

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 $\mathbf{x}$ 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 $\mathbf{x}$ 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 $\mathcal{Z} = \{\mathbf{x}_{z_1}, \ldots, \mathbf{x}_{z_k}\}$; then, the residual for a data point $\mathbf{x}$ encoded with k anchor points $f_1, \ldots, f_k$ is

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^k} \left\| \varphi(\mathbf{x}) - \sum_{j=1}^{k} \beta_j \varphi(\mathbf{x}_{z_j}) \right\|_{\mathcal{H}}^2,$$

which is equal to

$$\|\varphi(\mathbf{x})\|_{\mathcal{H}}^2 - \mathbf{f}_{\mathcal{Z}}(\mathbf{x})^\top \mathbf{K}_{\mathcal{Z}}^{-1} \mathbf{f}_{\mathcal{Z}}(\mathbf{x}),$$

and since $f_j = \varphi(\mathbf{x}_{z_j})$ for all j, the data point $\mathbf{x}_i$ with largest residual is the one that maximizes

$$K(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{f}_{\mathcal{Z}}(\mathbf{x}_i) \mathbf{K}_{\mathcal{Z}}^{-1} \mathbf{f}_{\mathcal{Z}}(\mathbf{x}_i) \;\; \text{with} \;\; \mathbf{f}_{\mathcal{Z}}(\mathbf{x}_i) = [K(\mathbf{x}_{z_1}, \mathbf{x}), \ldots, K(\mathbf{x}_{z_k}, \mathbf{x})]^\top.$$

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**

$$z_k \leftarrow \operatorname*{argmax}_{i \in \{1,\ldots,n\}} K(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{f}_{\mathcal{Z}}(\mathbf{x}_i)\mathbf{K}_{\mathcal{Z}}^{-1}\mathbf{f}_{\mathcal{Z}}(\mathbf{x}_i);$$

- **update the set $\mathcal{Z}$**

$$\mathcal{Z} \leftarrow \mathcal{Z} \cup \{\mathbf{x}_{z_k}\}.$$

Remarks

- A naive implementation costs $(O(k^2 n + k^3)$ 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 $\mathcal{Z}' = \mathcal{Z} \cup \{\mathbf{z}\}$,

$$\mathbf{K}_{\mathcal{Z}'}^{-1} = \left[\begin{array}{cc} \mathbf{K}_{\mathcal{Z}} & \mathbf{f}_{\mathcal{Z}}(\mathbf{z}) \\ \mathbf{f}_{\mathcal{Z}}(\mathbf{z})^\top & K(\mathbf{z},\mathbf{z}) \end{array} \right]^{-1} = \left[\begin{array}{cc} \mathbf{K}_{\mathcal{Z}}^{-1} + \frac{1}{s}\mathbf{b}\mathbf{b}^\top & -\frac{1}{s}\mathbf{b} \\ -\frac{1}{s}\mathbf{b}^\top & \frac{1}{s} \end{array} \right],$$

where s is the Schur complement $s = K(\mathbf{z},\mathbf{z}) - \mathbf{f}_{\mathcal{Z}}(\mathbf{z})\mathbf{K}_{\mathcal{Z}}^{-1}\mathbf{f}_{\mathcal{Z}}(\mathbf{z})$, and $\mathbf{b} = \mathbf{K}_{\mathcal{Z}}^{-1}\mathbf{f}_{\mathcal{Z}}(\mathbf{z})$.

Complexity analysis

- $\mathbf{K}_{\mathcal{Z}'}^{-1}$ can be obtained from $\mathbf{K}_{\mathcal{Z}}^{-1}$ and $\mathbf{f}_{\mathcal{Z}}(\mathbf{z})$ in $O(k^2)$ float operations; for that we need to always keep into memory the n vectors $\mathbf{f}_{\mathcal{Z}}(\mathbf{x}_i)$.
- updating the $\mathbf{f}_{\mathcal{Z}'}(\mathbf{x}_i)$'s from $\mathbf{f}_{\mathcal{Z}}(\mathbf{x}_i)$ requires n kernel evaluations;

The total training complexity is $O(p^2 n)$ 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 $\mathbf{z}_1, \ldots, \mathbf{z}_p$ such that they represented well some training data $\mathbf{x}_1, \ldots, \mathbf{x}_n$, 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 $\mathbf{z}_1, \ldots, \mathbf{z}_p$ in $\mathbb{R}^p$.
2. Define the anchor points $f_j = \varphi(\mathbf{z}_j)$ 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 $\mathbf{K} \approx \mathbf{L}\mathbf{L}^\top$;
- 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

Random Fourier features [Rahimi and Recht, 2007] (1/5)

A large class of approximations for shift-invariant kernels are based on sampling techniques. Consider a real-valued positive-definite continuous translation-invariant kernel $K(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x} - \mathbf{y})$ with $\kappa : \mathbb{R}^d \to \mathbb{R}$. Then, if $\kappa(0) = 1$, Bochner theorem tells us that κ is a valid characteristic function for some probability measure

$$\kappa(\mathbf{z}) = \mathbb{E}_{\mathbf{w}}[e^{i\mathbf{w}^\top \mathbf{z}}].$$

Remember indeed that, with the right assumptions on κ,

$$\kappa(\mathbf{x} - \mathbf{y}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{\kappa}(\mathbf{w}) e^{i\mathbf{w}^\top \mathbf{x}} e^{-i\mathbf{w}^\top \mathbf{y}} d\mathbf{w},$$

and the probability measure admits a density $q(\mathbf{w}) = \frac{1}{(2\pi)^d}\hat{\kappa}(\mathbf{w})$ (non-negative, real-valued, sum to 1 since $\kappa(0) = 1$).

Random Fourier features (2/5)

Then,

$$\begin{aligned}
\kappa(\mathbf{x}-\mathbf{y}) &= \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} \hat{\kappa}(\mathbf{w}) e^{i\mathbf{w}^\top \mathbf{x}} e^{-i\mathbf{w}^\top \mathbf{y}} d\mathbf{w} \\
&= \int_{\mathbb{R}^d} q(\mathbf{w}) \cos(\mathbf{w}^\top \mathbf{x} - \mathbf{w}^\top \mathbf{y}) d\mathbf{w} \\
&= \int_{\mathbb{R}^d} q(\mathbf{w}) \left(\cos(\mathbf{w}^\top \mathbf{x})\cos(\mathbf{w}^\top \mathbf{y}) + \sin(\mathbf{w}^\top \mathbf{x})\sin(\mathbf{w}^\top \mathbf{y})\right) d\mathbf{w} \\
&= \int_{\mathbb{R}^d}\int_{b=0}^{2\pi} \frac{q(\mathbf{w})}{2\pi} 2\cos(\mathbf{w}^\top \mathbf{x} + b)\cos(\mathbf{w}^\top \mathbf{y} + b) d\mathbf{w} db \quad \text{(exercise)} \\
&= \mathbb{E}_{\mathbf{w}\sim q(\mathbf{w}), b\sim \mathcal{U}[0,2\pi]}\left[\sqrt{2}\cos(\mathbf{w}^\top \mathbf{x} + b)\sqrt{2}\cos(\mathbf{w}^\top \mathbf{y} + b)\right]
\end{aligned}$$

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 $\varepsilon > 0$,

$$\mathbb{P}\left[\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} |\kappa(\mathbf{x}-\mathbf{y}) - \langle\psi(\mathbf{x}),\psi(\mathbf{y})\rangle_{\mathbb{R}^p}| \geq \varepsilon\right] \leq 2^8 \left(\frac{\sigma_q \mathrm{diam}(\mathcal{X})}{\varepsilon}\right)^2 e^{-\frac{p\varepsilon^2}{4(m+2)}},$$

where $\sigma_q^2 = \mathbb{E}_{\mathbf{w}\sim q(\mathbf{w})}[\mathbf{w}^\top \mathbf{w}]$ is the second moment of the Fourier transform of κ.

Remarks

- The convergence is uniform, not data dependent;
- Take the sequence $\varepsilon_p = \sqrt{\frac{\log(p)}{p}}\sigma_q \mathrm{diam}(\mathcal{X})$; 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 *fixed* pair of points $\mathbf{x}, \mathbf{y}$, Hoeffding's inequality says that

$$\mathbb{P}\big[\underbrace{|\kappa(\mathbf{x}-\mathbf{y}) - \langle\psi(\mathbf{x}), \psi(\mathbf{y})\rangle_{\mathbb{R}^d}|}_{f(\mathbf{x},\mathbf{y})} \geq \varepsilon\big] \leq 2e^{-\frac{p\varepsilon^2}{4}}.$$

- Consider a net (set of balls of radius r) that covers $\mathcal{X}_\Delta = \{\mathbf{x}-\mathbf{y} : (\mathbf{x}, \mathbf{y}) \in \mathcal{X}\}$ with at most $T = (4\text{diam}(\mathcal{X})/r)^m$ balls.
- Apply the Hoeffding's inequality to the centers $\mathbf{x}_i - \mathbf{y}_i$ of the balls;
- Use a basic union bound

$$\mathbb{P}\left[\sup_i f(\mathbf{x}_i, \mathbf{y}_i) \geq \frac{\varepsilon}{2}\right] \leq \sum_i \mathbb{P}\left[f(\mathbf{x}_i, \mathbf{y}_i) \geq \frac{\varepsilon}{2}\right] \leq 2Te^{-\frac{p\varepsilon^2}{8}}.$$

- Glue things together: control the probability for points $(\mathbf{x}, \mathbf{y})$ inside each ball, and adjust the radius r (a bit technical).

Outline

1. Kernels and RKHS

2. Kernel tricks

3. Kernel Methods: Supervised Learning

4. Kernel Methods: Unsupervised 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

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

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!

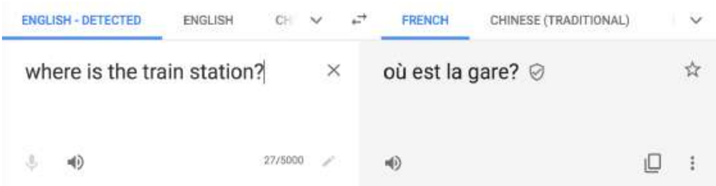
A functional space viewpoint

- 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 $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

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

What is specific to multilayer neural networks?

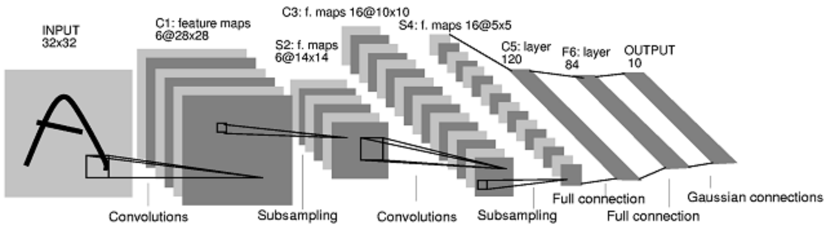
- The "neural network" space $\mathcal{F}$ is explicitly parametrized by:

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

- Linear operations are either unconstrained (fully connected) or involve parameter sharing (e.g., convolutions).
- Finding the optimal $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_k$ yields a **non-convex** optimization problem.

Convolutional Neural Networks

Picture from LeCun et al. (1998)

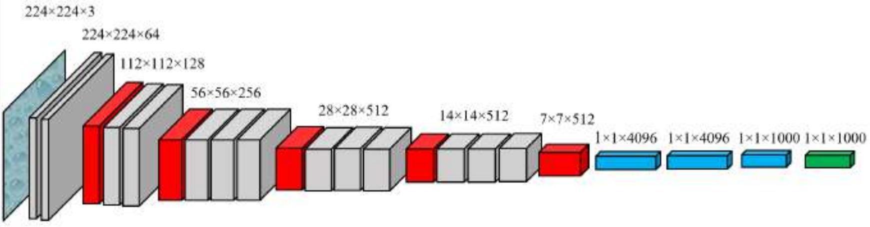


What are the main features of CNNs?

- they capture **compositional** and **multiscale** structures in images;
- they provide some **invariance**;
- they model the **local stationarity** of images at several scales;

Convolutional Neural Networks

(Simonyan and Zisserman, 2014)



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.

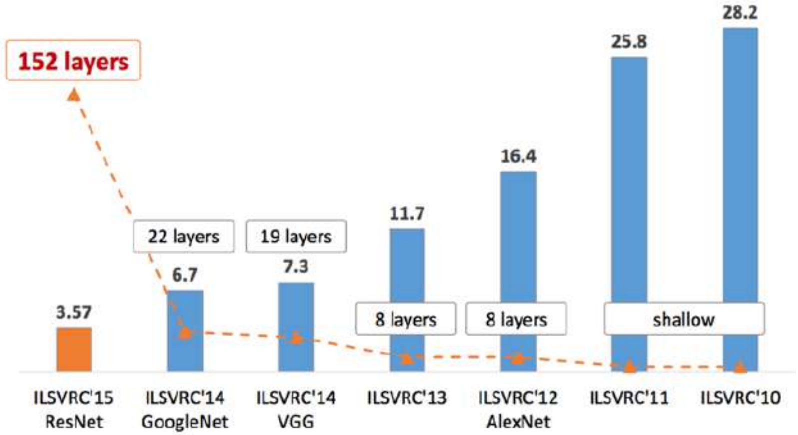


Figure: Top-5 error rate

Convolutional neural networks for biological sequences

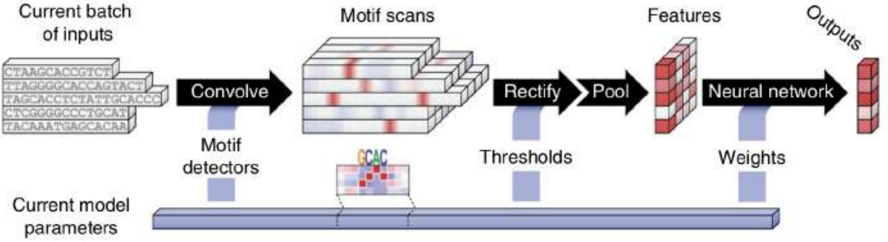


Figure: two-layer CNN architecture from Alipanahi et al. (2015)

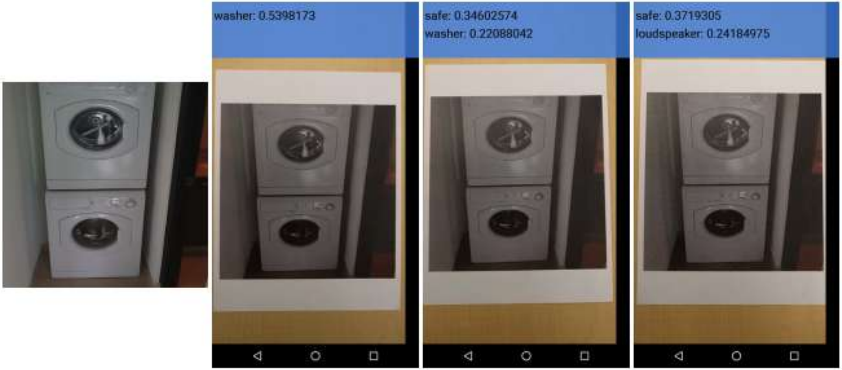
- Sequences are represented by one-hot encoding (A=(1,0,0,0),C=(0,1,0,0),. . .).
- Single convolution layer followed by linear classifier.

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. . . .

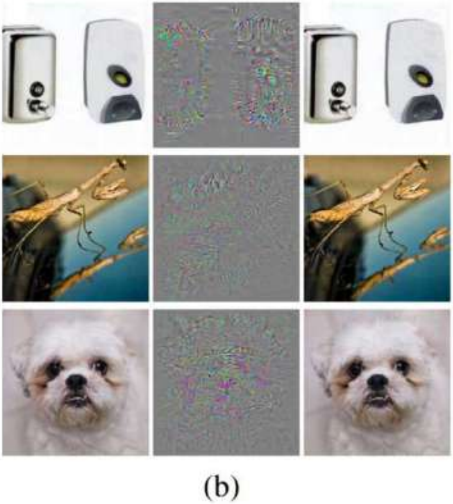
Adversarial examples, Picture from Kurakin et al. (2016)



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Figure: Adversarial examples are generated by computer; then printed on paper; a new picture taken on a smartphone fools the classifier.

Adversarial examples

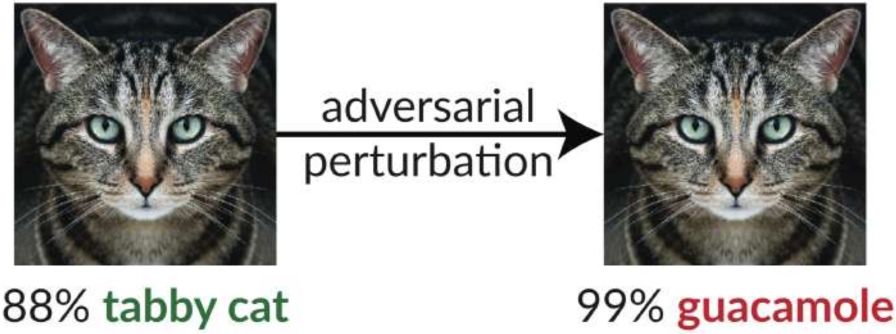


clean + noise → "**ostrich**" (Szegedy et al., 2013).

Adversarial examples

(a real ostrich)

Adversarial examples



`https://github.com/anishathalye/obfuscated-gradients`

Convolutional Neural Networks

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

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**?

$$|f(\mathbf{x}) - f(\mathbf{x}')| \text{ should be close if } \mathbf{x} \text{ and } \mathbf{x}' \text{ are "similar".}$$

- what does it mean for x and x' to be "similar"?
- what should be a good **regularization function** Ω?

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

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.

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?

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

$$K(x, x') = \kappa_2(\langle \varphi_1(x), \varphi_1(x') \rangle) = \kappa_2(\kappa_1(x^\top x'))$$

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 $K : \mathcal{X}^2 \rightarrow \mathbb{R}$

$$K(\mathbf{x}, \mathbf{y}) = \kappa(\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}_0}),$$

is positive definite for all $\mathcal{H}_0$ if and only if κ 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

linear kernel	$\langle z, z' \rangle$
exponential kernel	$e^{\alpha(\langle z,z' \rangle - 1)}$
inverse polynomial kernel	$\frac{1}{2-\langle z,z' \rangle}$
polynomial kernel of degree p	$(c + \langle z, z' \rangle)^p$
arc-cosine kernel of degree 1	$\frac{1}{\pi}\left(\sin(\theta) + (\pi - \theta)\cos(\theta)\right)$ 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

linear kernel	$\langle z, z' \rangle$
exponential kernel	$e^{\alpha(\langle z,z' \rangle - 1)}$
inverse polynomial kernel	$\frac{1}{2-\langle z,z' \rangle}$
polynomial kernel of degree p	$(c + \langle z, z' \rangle)^p$
arc-cosine kernel of degree 1	$\frac{1}{\pi}\left(\sin(\theta) + (\pi - \theta)\cos(\theta)\right)$ with $\theta = \arccos(\langle z, z' \rangle)$
Vovk's kernel of degree 3	$\frac{1}{3}\left(\frac{1-\langle z,z' \rangle^3}{1-\langle z,z' \rangle}\right) = \frac{1}{3}\left(1 + \langle z, z' \rangle + \langle z, z' \rangle^2\right)$

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

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \qquad m \to \infty$$

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

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \qquad m \to \infty$$

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

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \qquad m \to \infty$$

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

$$\begin{cases} J_0(\theta) & = \pi - \theta \\ J_1(\theta) & = \sin(\theta) + (\pi - \theta)\cos(\theta) \\ J_2(\theta) & = 3\sin(\theta)\cos(\theta) + (\pi - \theta)(1 + 2\cos^2(\theta)) \end{cases}$$

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

$$k_n(x,y) = \mathbb{E}_{w\sim N(0,I)}[\sigma(w^\top x)\sigma(w^\top y)],$$

with $\sigma(u) = \frac{u^n}{\sqrt{2}}(1+\text{sign}(u))$.

- Note that $k_1(x,y) = \mathbb{E}_{w\sim N(0,I)}[\text{RELU}(w^\top x)\text{RELU}(w^\top y)]$.
- One of the fundamental tool to analyze RELU networks.

Kernels for deep models: neural tangent kernels

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \qquad m \to \infty$$

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): θ stays close to θ_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**

Kernels for deep models: neural tangent kernels

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \qquad m \to \infty$$

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): θ stays close to θ_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

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x), \qquad m \to \infty$$

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): θ stays close to θ_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**

$$K_{NTK}(x, y) = \mathbb{E}_{\mathbf{w}}[\sigma(\mathbf{w}^\top x)\sigma(\mathbf{w}^\top y) + (x^\top y)\sigma'(\mathbf{w}^\top x)\sigma'(\mathbf{w}^\top y)]$$

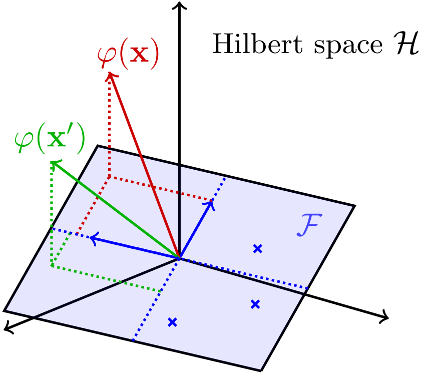
- with RELU networks, we obtain a dot-product kernel.

Kernels for deep models: dot-product kernels + Nyström

The Nyström method consists of replacing any point $\varphi(\mathbf{x})$ in $\mathcal{H}$, for $\mathbf{x}$ in $\mathcal{X}$ by its orthogonal projection onto a finite-dimensional subspace

$$\mathcal{F} = \text{span}(\varphi(\mathbf{z}_1), \ldots, \varphi(\mathbf{z}_p)),$$

for some anchor points $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_p]$ in $\mathbb{R}^{d \times p}$



Kernels for deep models: dot-product kernels + Nyström

The projection is equivalent to

$$\Pi_{\mathcal{F}}[\mathbf{x}] := \sum_{j=1}^{p} \beta_j^\star \varphi(\mathbf{z}_j) \quad \text{with} \quad \boldsymbol{\beta}^\star \in \operatorname*{argmin}_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\| \varphi(\mathbf{x}) - \sum_{j=1}^{p} \beta_j \varphi(\mathbf{z}_j) \right\|_{\mathcal{H}}^2,$$

Then, it is possible to show that with $K(\mathbf{x}, \mathbf{y}) = \kappa(\langle \mathbf{x}, \mathbf{y} \rangle)$,

$$K(\mathbf{x}, \mathbf{y}) \approx \langle \Pi_{\mathcal{F}}[\mathbf{x}], \Pi_{\mathcal{F}}[\mathbf{y}] \rangle_{\mathcal{H}} = \langle \psi(\mathbf{x}), \psi(\mathbf{y}) \rangle_{\mathbb{R}^p},$$

with

$$\psi(\mathbf{x}) = \kappa(\mathbf{Z}^\top \mathbf{Z})^{-1/2} \kappa(\mathbf{Z}^\top \mathbf{x}),$$

where the function κ is applied pointwise to its arguments. The resulting ψ 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).

Kernels for deep models: end-to-end learning

Nyström's encoding with a dot-product kernel provides the encoding

$$\psi_{\mathbf{Z}}(\mathbf{x}) = \kappa(\mathbf{Z}^\top \mathbf{Z})^{-1/2} \kappa(\mathbf{Z}^\top \mathbf{x}).$$

The anchor points **Z** can be learned in various manners

- **unsupervised learning**: use K-means!
- **supervised learning**: use back-propagation

$$\min_{\mathbf{w},\mathbf{Z}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \mathbf{w}^\top \psi_{\mathbf{Z}}(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|^2.$$

Kernels for deep models: end-to-end learning

Nyström's encoding with a dot-product kernel provides the encoding

$$\psi_{\mathbf{Z}}(\mathbf{x}) = \kappa(\mathbf{Z}^\top \mathbf{Z})^{-1/2} \kappa(\mathbf{Z}^\top \mathbf{x}).$$

The anchor points **Z** can be learned in various manners

- **unsupervised learning**: use K-means!
- **supervised learning**: use back-propagation

$$\min_{\mathbf{w}, \mathbf{Z}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \mathbf{w}^\top \psi_{\mathbf{Z}}(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|^2.$$

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).

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k \sigma_{k-1}(\mathbf{A}_{k-1} \ldots \sigma_2(\mathbf{A}_2 \sigma_1(\mathbf{A}_1 \mathbf{x})) \ldots)) = \langle f, \Phi(x) \rangle_{\mathcal{H}}.$$

- 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).

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k\sigma_{k-1}(\mathbf{A}_{k-1}\ldots\sigma_2(\mathbf{A}_2\sigma_1(\mathbf{A}_1\mathbf{x}))\ldots)) = \langle f, \Phi(x)\rangle_{\mathcal{H}}.$$

- we call the construction "**convolutional kernel networks**".

Simple story about CKNs (Mairal, 2016)

- for the theory part, replace $\mathbf{x} \mapsto \sigma(\mathbf{A}\mathbf{x})$ at each CNN layer by a kernel mapping $\mathbf{x} \mapsto \varphi(\mathbf{x})$ associated to a dot-product kernel.
- for the practical part, replace $\mathbf{x} \mapsto \sigma(\mathbf{A}\mathbf{x})$ by Nyström's embedding $\mathbf{x} \mapsto \kappa(\mathbf{Z}^\top\mathbf{Z})^{-1/2}\kappa(\mathbf{Z}^\top\mathbf{x})$. 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).

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k\sigma_{k–1}(\mathbf{A}_{k–1}\ldots\sigma_2(\mathbf{A}_2\sigma_1(\mathbf{A}_1\mathbf{x}))\ldots)) = \langle f, \Phi(x)\rangle_{\mathcal{H}}.$$

- 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 $L^2(\Omega, \mathcal{H}_0)$

$x_0 : \Omega \to \mathcal{H}_0$: **continuous** signal, with $\Omega = \mathbb{R}^d$ ($d = 2$ for images).

- $x_0(u) \in \mathcal{H}_0$: input value at location u $\quad$ ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images).

Construction of the RKHS for continuous signals

Initial map x_0 in $L^2(\Omega, \mathcal{H}_0)$

$x_0 : \Omega \to \mathcal{H}_0$: **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 $L^2(\Omega, \mathcal{H}_{k-1})$

$x_k : \Omega \to \mathcal{H}_k$: **feature map** at layer 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).

Construction of the RKHS for continuous signals

Initial map x_0 in $L^2(\Omega, \mathcal{H}_0)$

$x_0 : \Omega \to \mathcal{H}_0$: **continuous** signal, with $\Omega = \mathbb{R}^d$ ($d = 2$ for images).

- $x_0(u) \in \mathcal{H}_0$: input value at location u $\quad$ ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images).

Building map x_k in $L^2(\Omega, \mathcal{H}_k)$ from x_{k-1} in $L^2(\Omega, \mathcal{H}_{k-1})$

$x_k : \Omega \to \mathcal{H}_k$: **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 $L^2(\Omega, \mathcal{H}_0)$

$x_0 : \Omega \to \mathcal{H}_0$: **continuous** signal, with $\Omega = \mathbb{R}^d$ ($d = 2$ for images).

- $x_0(u) \in \mathcal{H}_0$: input value at location u $\quad (\mathcal{H}_0 = \mathbb{R}^3$ for RGB images).

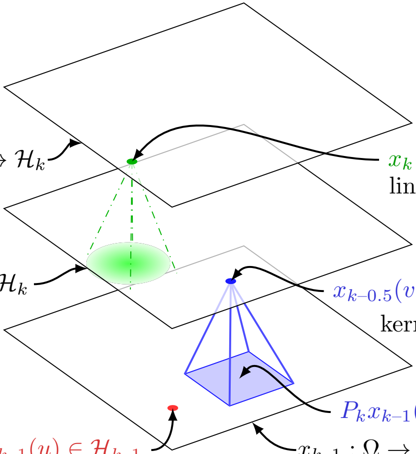
Building map x_k in $L^2(\Omega, \mathcal{H}_k)$ from x_{k-1} in $L^2(\Omega, \mathcal{H}_{k-1})$

$x_k : \Omega \to \mathcal{H}_k$: **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 σ_k.

Construction of the RKHS for continuous signals



$x_k : \Omega \to \mathcal{H}_k$

$x_k(w) \in \mathcal{H}_k$

linear pooling

$x_{k-0.5} : \Omega \to \mathcal{H}_k$

$x_{k-0.5}(v) = \varphi_k(P_k x_{k-1}(v)) \in \mathcal{H}_k$

kernel mapping

$P_k x_{k-1}(v) \in \mathcal{P}_k$ (patch extraction)

$x_{k-1}(u) \in \mathcal{H}_{k-1}$

$x_{k-1} : \Omega \to \mathcal{H}_{k-1}$

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

$$\Phi_n(x) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \ \in \ L^2(\Omega, \mathcal{H}_n).$$

- σ_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

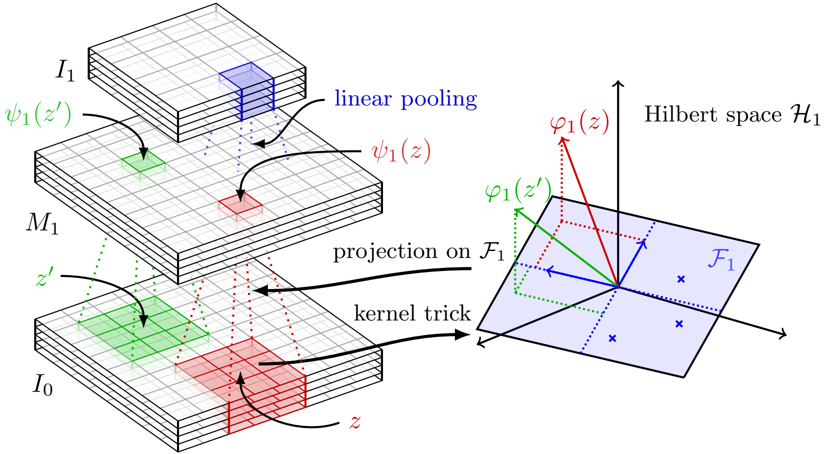
$$\Phi_n(x) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \ \in \ L^2(\Omega, \mathcal{H}_n).$$

- σ_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" $\mathcal{K}(x, x') = \langle \Phi_n(x), \Phi_n(x')\rangle = \int_\Omega \langle x_n(u), x'_n(u)\rangle du$.

Convolutional Kernel Networks in practice



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 $\psi_{CNN}(\mathbf{x}) = \sigma(\mathbf{Z}^\top \mathbf{x})$.
- whereas a CKN computes $\psi_{CKN}(\mathbf{x}) = \|\mathbf{x}\| \kappa(\mathbf{Z}^\top \mathbf{Z})^{-1/2} \kappa(\mathbf{Z}^\top \mathbf{x}/\|\mathbf{x}\|)$.

Convolutional Kernel Networks in Practice

What is the difference with a CNN?

- Given a patch x, a CNN computes $\psi_{CNN}(\mathbf{x}) = \sigma(\mathbf{Z}^\top \mathbf{x})$.
- whereas a CKN computes $\psi_{CKN}(\mathbf{x}) = \|\mathbf{x}\| \kappa(\mathbf{Z}^\top \mathbf{Z})^{-1/2} \kappa(\mathbf{Z}^\top \mathbf{x} / \|\mathbf{x}\|)$.

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

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

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).

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k\sigma_{k\text{–}1}(\mathbf{A}_{k\text{–}1} \ldots \sigma_2(\mathbf{A}_2\sigma_1(\mathbf{A}_1\mathbf{x}))\ldots)) = \langle f, \Phi(x)\rangle_{\mathcal{H}}.$$

- 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?

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).

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k\sigma_{k-1}(\mathbf{A}_{k-1}\ldots\sigma_2(\mathbf{A}_2\sigma_1(\mathbf{A}_1\mathbf{x}))\ldots)) = \langle f, \Phi(x)\rangle_{\mathcal{H}}.$$

- 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 Φ.
- **Stability to deformations and non-expansiveness** for Φ.
- 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 Φ.
- **Stability to deformations and non-expansiveness** for Φ.
- Constructions to achieve **group invariance**.

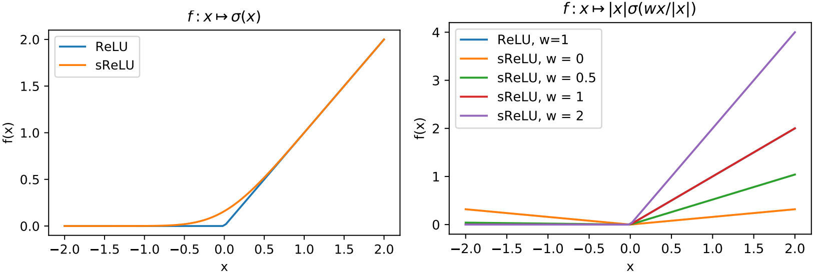
On learning

- Bounds on the RKHS norm $\|.\|_{\mathcal{H}}$ 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

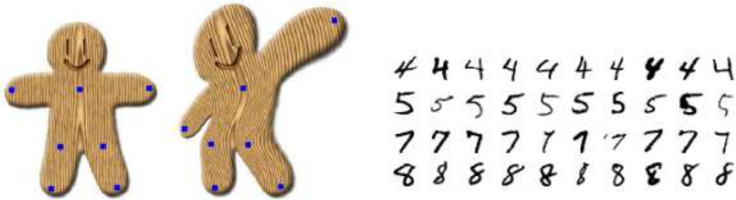
$$z \mapsto \mathrm{ReLU}(w^\top z) \qquad \Longrightarrow \qquad z \mapsto \|z\|\sigma\big(w^\top z/\|z\|\big).$$



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



- 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: $f(x) = \langle f, \Phi(x) \rangle$

$$|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \cdot \|\Phi(x) - \Phi(x')\|_{\mathcal{H}}$$

- $\|f\|_{\mathcal{H}}$ 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: $f(x) = \langle f, \Phi(x) \rangle$

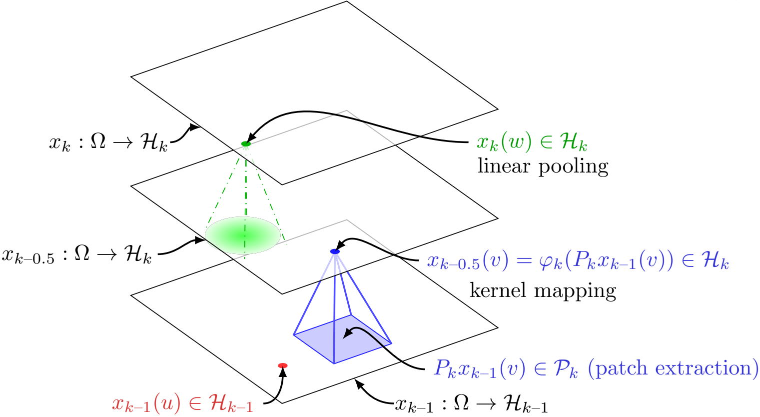
$$|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \cdot \|\Phi(x) - \Phi(x')\|_{\mathcal{H}}$$

- $\|f\|_{\mathcal{H}}$ 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)

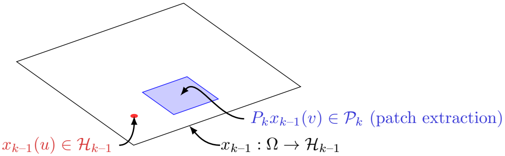
Recap: Construction of the RKHS for continuous signals



$$\Phi_n(x) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \;\in\; L^2(\Omega, \mathcal{H}_n).$$

Patch extraction operator P_k

$$P_k x_{k-1}(u) := (x_{k-1}(u+v))_{v \in S_k} \in \mathcal{P}_k = \mathcal{H}_{k-1}^{S_k}$$



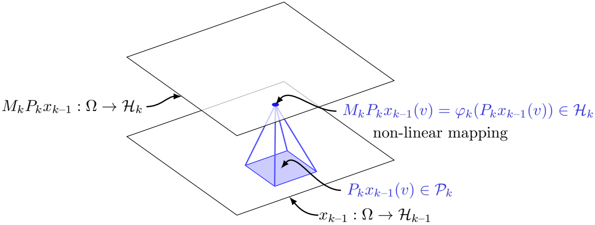
Patch extraction operator P_k

$$P_k x_{k-1}(u) := (x_{k-1}(u+v))_{v \in S_k} \in \mathcal{P}_k = \mathcal{H}_{k-1}^{S_k}$$

- 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

$$M_k P_k x_{k-1}(u) := \varphi_k(P_k x_{k-1}(u)) \in \mathcal{H}_k$$

Kernel mapping of **homogeneous dot-product kernels**:

$$K_k(z, z') = \|z\| \|z'\| \kappa_k \left(\frac{\langle z, z' \rangle}{\|z\| \|z'\|} \right) = \langle \varphi_k(z), \varphi_k(z') \rangle.$$

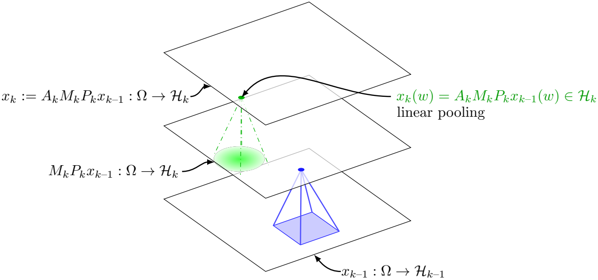
$\kappa_k(u) = \sum_{j=0}^{\infty} b_j u^j$ with $b_j \geq 0$, $\kappa_k(1) = 1$

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

$$x_k(u) = A_k M_k P_k x_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u-v) M_k P_k x_{k-1}(v) dv \in \mathcal{H}_k$$



Pooling operator A_k

$$x_k(u) = A_k M_k P_k x_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u-v) M_k P_k x_{k-1}(v) dv \in \mathcal{H}_k$$

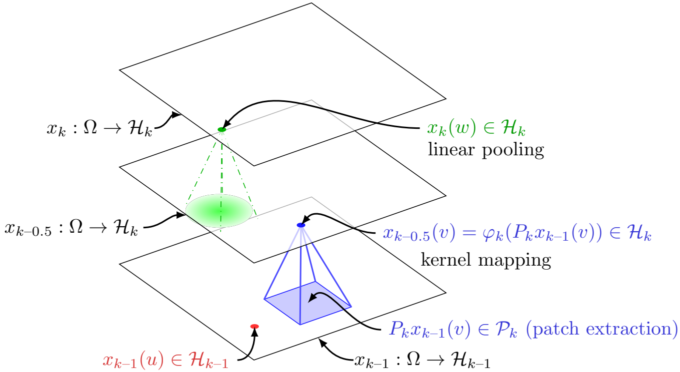
- h_{σ_k}: pooling filter at scale σ_k
- $h_{\sigma_k}(u) := \sigma_k^{-d} h(u/\sigma_k)$ with $h(u)$ **Gaussian**
- **linear, non-expansive operator**: $\|A_k\| \leq 1$

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_{σ_k}: pooling filter at scale σ_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 σ_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, σ_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, σ_k grow exponentially in practice (i.e., fixed with subsampling).

Assumption on x_0

- x_0 is typically a **discrete** signal aquired with physical device.
- Natural assumption: $x_0 = A_0 x$, with x the original continuous signal, A_0 local integrator with scale σ_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, σ_k grow exponentially in practice (i.e., fixed with subsampling).

Assumption on x_0

- x_0 is typically a **discrete** signal aquired with physical device.
- Natural assumption: $x_0 = A_0 x$, with x the original continuous signal, A_0 local integrator with scale σ_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: $\Box L_c = L_c \Box$.

$$\begin{aligned} \|\Phi_n(L_c x) - \Phi_n(x)\| &= \|L_c \Phi_n(x) - \Phi_n(x)\| \\ &\leq \|L_c A_n - A_n\| \cdot \|M_n P_n \Phi_{n-1}(x)\| \\ &\leq \|L_c A_n - A_n\| \|x\|. \end{aligned}$$

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: $\square L_c = L_c \square$.

$$\begin{aligned} \|\Phi_n(L_c x) - \Phi_n(x)\| &= \|L_c \Phi_n(x) - \Phi_n(x)\| \\ &\leq \|L_c A_n - A_n\| \cdot \|M_n P_n \Phi_{n-1}(x)\| \\ &\leq \|L_c A_n - A_n\| \|x\|. \end{aligned}$$

- 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: $\square L_c = L_c \square$.

$$\begin{aligned} \|\Phi_n(L_c x) - \Phi_n(x)\| &= \|L_c \Phi_n(x) - \Phi_n(x)\| \\ &\leq \|L_c A_n - A_n\| \cdot \|M_n P_n \Phi_{n-1}(x)\| \\ &\leq \|L_c A_n - A_n\| \|x\|. \end{aligned}$$

- Mallat (2012): $\|L_c A_n - A_n\| \leq \frac{C_2}{\sigma_n} c$ (operator norm).
- **Scale σ_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_τ!

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_τ!
- $\|A_k L_\tau - L_\tau A_k\| \leq C_1 \|\nabla \tau\|_\infty$ (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_τ!
- $\|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$ (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_τ!
- $\|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$ (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_τ!
- $\|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$ (from Mallat, 2012).
- But: $[P_k, L_\tau]$ is **unstable** at high frequencies!
- Adapt to **current layer resolution**, patch size controlled by σ_{k-1}:

$$\|[P_k A_{k-1}, L_\tau]\| \leq C_{1,\kappa} \|\nabla \tau\|_\infty \qquad \sup_{u \in S_k} |u| \leq \kappa \sigma_{k-1}$$

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_τ!
- $\|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$ (from Mallat, 2012).
- But: $[P_k, L_\tau]$ is **unstable** at high frequencies!
- Adapt to **current layer resolution**, patch size controlled by σ_{k-1}:

$$\|[P_k A_{k-1}, L_\tau]\| \leq C_{1,\kappa} \|\nabla \tau\|_\infty \qquad \sup_{u \in S_k} |u| \leq \kappa \sigma_{k-1}$$

- $C_{1,\kappa}$ grows as $\kappa^{d+1} \implies$ more stable with **small patches** (e.g., 3x3, VGG et al.).

Stability to deformations

Theorem (Stability of CKN (Bietti and Mairal, 2019a))

Let $\Phi_n(x) = \Phi(A_0 x)$ *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 σ_n
- Stability: small patch sizes ($\beta \approx$ patch size, $C_\beta = O(\beta^3)$ for images)
- Signal preservation: subsampling factor $\approx$ patch size

$\Longrightarrow$ need several layers with small patches $n = O(\log(\sigma_n/\sigma_0)/\log\beta)$

Stability to deformations

Theorem (Stability of CKN (Bietti and Mairal, 2019a))

Let $\Phi_n(x) = \Phi(A_0 x)$ *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 σ_n
- Stability: small patch sizes ($\beta \approx$ patch size, $C_\beta = O(\beta^3)$ for images)
- Signal preservation: subsampling factor $\approx$ patch size

$\implies$ need several layers with small patches $n = O(\log(\sigma_n/\sigma_0)/\log\beta)$

- 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 σ_{k-1}: 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**:

$$Px(u) = (x(uv))_{v \in S}.$$

- **Non-linear mapping**: equivariant because pointwise!
- **Pooling** (μ: left-invariant Haar measure):

$$Ax(u) = \int_G x(uv)h(v)d\mu(v) = \int_G x(v)h(u^{-1}v)d\mu(v).$$

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* $\|\nabla\tau\|_\infty \leq 1/2$

$$\begin{aligned}&\|\Phi_n(L_\tau x) - \Phi_n(x)\| \\ &\quad \leq \left(C_\beta n^{7/4} \|\nabla\tau\|_\infty^{1/2} + C'_\beta n^2 \|\nabla\tau\|_\infty + \sqrt{n+1}\frac{C}{\sigma_n}\|\tau\|_\infty \right) \|x\|,\end{aligned}$$

Discretization and signal preservation: example in 1D

- Discrete signal $\bar{x}_k$ in $\ell^2(\mathbb{Z}, \bar{\mathcal{H}}_k)$ vs continuous ones x_k in $L^2(\mathbb{R}, \mathcal{H}_k)$.
- $\bar{x}_k$: subsampling factor s_k after pooling with scale $\sigma_k \approx s_k$:

$$\bar{x}_k[n] = \bar{A}_k \bar{M}_k \bar{P}_k \bar{x}_{k-1}[ns_k].$$

Discretization and signal preservation: example in 1D

- Discrete signal $\bar{x}_k$ in $\ell^2(\mathbb{Z}, \bar{\mathcal{H}}_k)$ 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 $\ell^2(\mathbb{Z}, \bar{\mathcal{H}}_k)$ vs continuous ones x_k in $L^2(\mathbb{R}, \mathcal{H}_k)$.
- $\bar{x}_k$: subsampling factor s_k after pooling with scale $\sigma_k \approx s_k$:

$$\bar{x}_k[n] = \bar{A}_k \bar{M}_k \bar{P}_k \bar{x}_{k-1}[n s_k].$$

- 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 $\bar{\mathcal{H}}_k$)

$$\langle f_w, \bar{M}_k \bar{P}_k \bar{x}_{k-1}(u) \rangle = f_w(\bar{P}_k \bar{x}_{k-1}(u)) = \langle w, \bar{P}_k \bar{x}_{k-1}(u) \rangle,$$

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 $\ell^2(\mathbb{Z}, \bar{\mathcal{H}}_k)$ 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**.
- How? Recover patches with **linear functions** (contained in $\bar{\mathcal{H}}_k$)

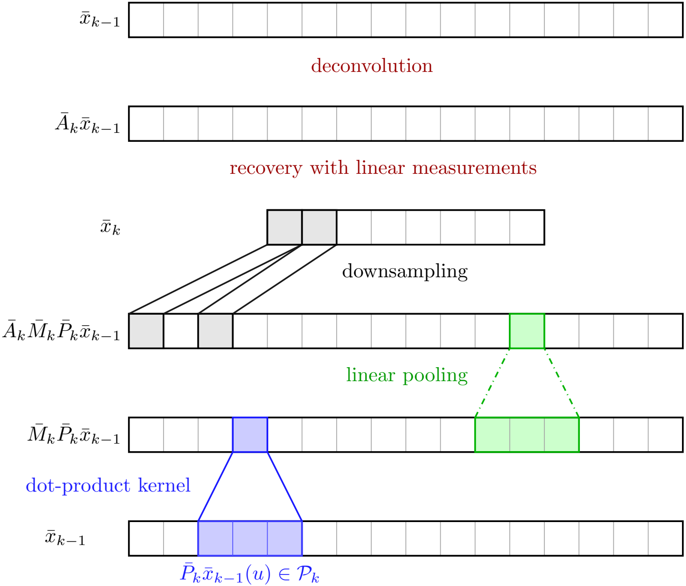
$$\langle f_w, \bar{M}_k \bar{P}_k \bar{x}_{k-1}(u) \rangle = f_w(\bar{P}_k \bar{x}_{k-1}(u)) = \langle w, \bar{P}_k \bar{x}_{k-1}(u) \rangle,$$

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



RKHS of patch kernels K_k

$$K_k(z, z') = \|z\|\|z'\|\kappa\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right), \qquad \kappa(u) = \sum_{j=0}^{\infty} b_j u^j.$$

What does the RKHS contain?

Homogeneous version of (Zhang et al., 2016, 2017)

RKHS of patch kernels K_k

$$K_k(z,z') = \|z\|\|z'\|\kappa\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right), \qquad \kappa(u) = \sum_{j=0}^{\infty} b_j u^j.$$

What does the RKHS contain?

- RKHS contains **homogeneous functions**:

$$f : z \mapsto \|z\|\sigma(\langle g, z\rangle/\|z\|).$$

Homogeneous version of (Zhang et al., 2016, 2017)

RKHS of patch kernels K_k

$$K_k(z,z') = \|z\|\|z'\|\kappa\left(\frac{\langle z,z'\rangle}{\|z\|\|z'\|}\right), \qquad \kappa(u) = \sum_{j=0}^{\infty} b_j u^j.$$

What does the RKHS contain?

- RKHS contains **homogeneous functions**:

$$f : z \mapsto \|z\|\sigma(\langle g,z\rangle/\|z\|).$$

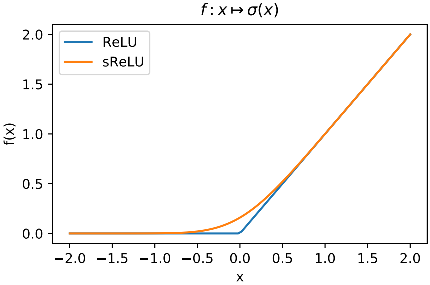
- **Smooth activations**: $\sigma(u) = \sum_{j=0}^{\infty} a_j u^j$ with $a_j \geq 0$.
- Norm: $\|f\|_{\mathcal{H}_k}^2 \leq C_\sigma^2(\|g\|^2) = \sum_{j=0}^{\infty} \frac{a_j^2}{b_j}\|g\|^2 < \infty$.

Homogeneous version of (Zhang et al., 2016, 2017)

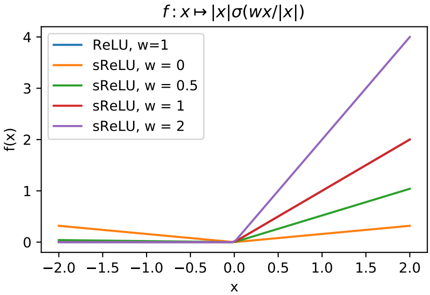
RKHS of patch kernels K_k

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: $C_\sigma^2(\lambda^2) = O(e^{c\lambda^2})$.



$f : x \mapsto \sigma(x)$



$f : x \mapsto |x|\sigma(wx/|x|)$

Constructing a CNN in the RKHS $\mathcal{H}_\mathcal{K}$

Some CNNs live in the RKHS: “linearization” principle

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k \sigma_{k-1}(\mathbf{A}_{k-1} \ldots \sigma_2(\mathbf{A}_2 \sigma_1(\mathbf{A}_1 \mathbf{x})) \ldots)) = \langle f, \Phi(x) \rangle_{\mathcal{H}}.$$

Constructing a CNN in the RKHS $\mathcal{H}_{\mathcal{K}}$

Some CNNs live in the RKHS: “linearization” principle

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k\sigma_{k-1}(\mathbf{A}_{k-1}\ldots\sigma_2(\mathbf{A}_2\sigma_1(\mathbf{A}_1\mathbf{x}))\ldots)) = \langle f, \Phi(x)\rangle_{\mathcal{H}}.$$

- Consider a CNN with filters $W_k^{ij}(u), u \in S_k$.
  - k: layer;
  - i: index of filter;
  - j: index of input channel.
- “Smooth homogeneous” activations σ.
- The CNN can be constructed hierarchically in $\mathcal{H}_{\mathcal{K}}$.
- 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:

$$\mathcal{F}_B = \{f \in \mathcal{H}_\mathcal{K}, \|f\| \leq B\} \implies \mathrm{Rad}_N(\mathcal{F}_B) \leq O\left(\frac{BR}{\sqrt{N}}\right).$$

Link with generalization

Direct application of classical generalization bounds

- Simple bound on Rademacher complexity for linear/kernel methods:

$$\mathcal{F}_B = \{f \in \mathcal{H}_{\mathcal{K}}, \|f\| \leq B\} \implies \mathrm{Rad}_N(\mathcal{F}_B) \leq O\left(\frac{BR}{\sqrt{N}}\right).$$

- 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 $\leq$ 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.

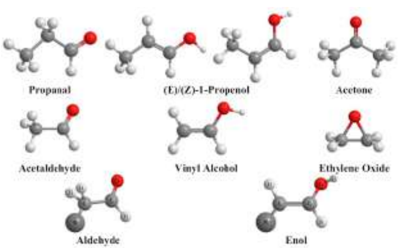
Questions:

- 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

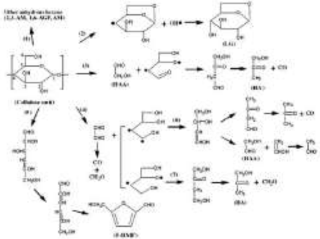
Graph-structured data is everywhere



(a) molecules

(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

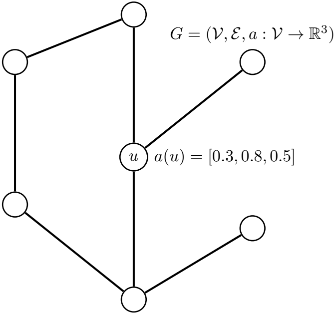
State-of-the-art models for representing graphs:

- **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:

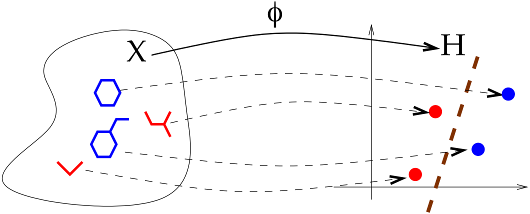
- A new type of **multilayer** graph kernel: more **expressive** than WL kernels;
- Learning easy-to-regularize and scalable **unsupervised** graph representations;
- Learning **supervised** graph representations like GNNs.

Graphs with node attributes



- 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;
- $a : \mathcal{V} \to \mathbb{R}^d$ is a function assigning attributes to each node.

Graph kernel mappings

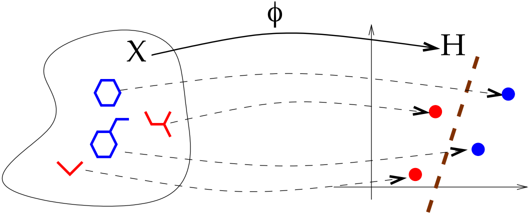


- Map each graph G in $\mathcal{X}$ to a vector $\Phi(G)$ in $\mathcal{H}$, which lends itself to learning tasks.
- A large class of graph kernel mappings can be written in the form

$$\Phi(G) := \sum_{u \in \mathcal{V}} \varphi_{\text{base}}(\ell_G(u)) \quad \text{where } \varphi_{\text{base}} \text{ embeds some local patterns } \ell$$

(Shervashidze et al., 2011; Lei et al., 2017; Kriege et al., 2019)

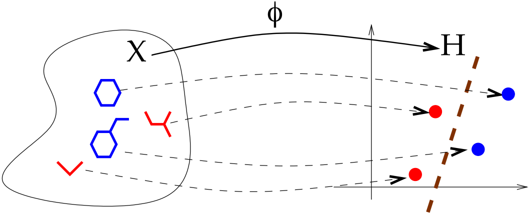
Graph kernel mappings



- 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

$$K(G, G') = \left\langle \underbrace{\sum_{u \in \mathcal{V}} \varphi_{\text{base}}(\ell_G(u))}_{\Phi(G)}, \underbrace{\sum_{u' \in \mathcal{V}'} \varphi_{\text{base}}(\ell_{G'}(u'))}_{\Phi(G')} \right\rangle .$$

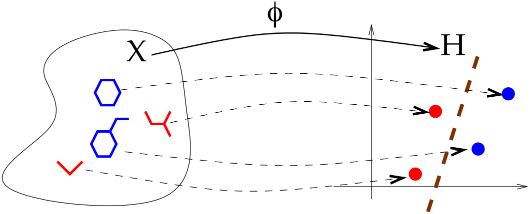
Graph kernel mappings



- 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

$$K(G, G') = \sum_{u \in \mathcal{V}} \sum_{u' \in \mathcal{V}'} \langle \varphi_{\text{base}}(\ell_G(u)), \varphi_{\text{base}}(\ell_{G'}(u')) \rangle.$$

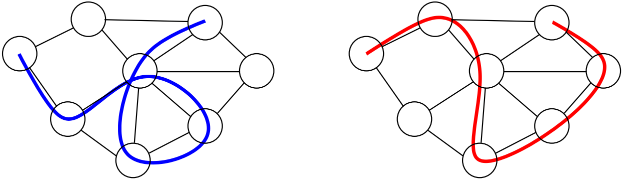
Graph kernel mappings



- 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

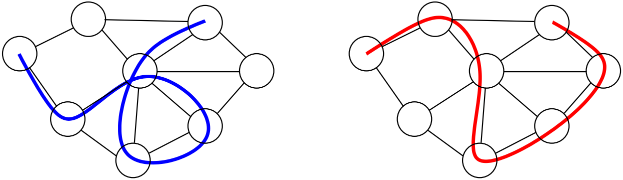
$$K(G, G') = \sum_{u \in \mathcal{V}} \sum_{u' \in \mathcal{V}'} \kappa_{\text{base}}(\ell_G(u), \ell_{G'}(u')).$$

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

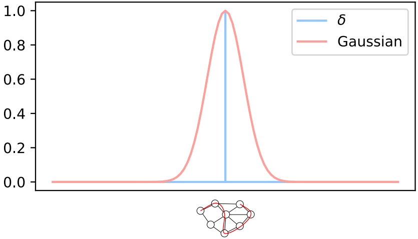


- $\mathcal{P}_k(G, u)$:= paths of length k from node u in G. The **k-path** mapping is

$$\varphi_{\text{path}}(u) := \sum_{p \in \mathcal{P}_k(G,u)} \delta_{a(p)} \quad \Longrightarrow \quad \Phi(G) = \sum_{u \in \mathcal{V}} \sum_{p \in \mathcal{P}_k(G,u)} \delta_{a(p)}.$$

- $a(p)$: concatenated attributes in p; δ: the Dirac function;
- $\Phi(G)$ can be interpreted as a **histogram** of paths occurrences;

A relaxed path kernel

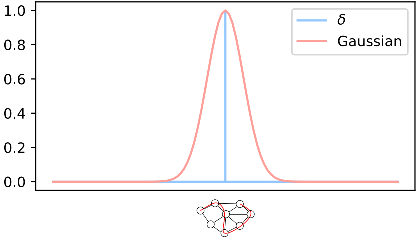


$$\varphi_{\text{path}}(u) = \sum_{p \in \mathcal{P}_k(G,u)} \delta_{a(p)}(\cdot)$$

Issues of the path kernel mapping:

- δ allows hard comparison between paths thus only works for discrete attributes;
- δ is not differentiable, which cannot be “optimized” with back-propagation.

A relaxed path kernel



$$\varphi_{\text{path}}(u) = \sum_{p \in \mathcal{P}_k(G,u)} \delta_{a(p)}(\cdot)$$

$$\implies \sum_{p \in \mathcal{P}_k(G,u)} e^{-\frac{\alpha}{2}\|a(p)-\cdot\|^2}.$$

Issues of the path kernel mapping:

- δ allows hard comparison between paths thus only works for discrete attributes;
- δ 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.

One-layer GCKN: a closer look at the relaxed path kernel

- We define the one-layer GCKN as the relaxed path kernel mapping

$$\varphi_1(u) := \sum_{p \in \mathcal{P}_k(G,u)} e^{-\frac{\alpha_1}{2}\|a(p)-\cdot\|^2} = \sum_{p \in \mathcal{P}_k(G,u)} \varphi_{\mathrm{RBF}}(a(p)) \in \mathcal{H}_1.$$

- This formula can be divided into **3 steps**:
  - path extraction: enumerating all $\mathcal{P}_k(G,u)$;
  - kernel mapping: evaluating Gaussian embedding φ_{RBF} of path features;
  - path aggregation: aggregating the path embeddings.

One-layer GCKN: a closer look at the relaxed path kernel

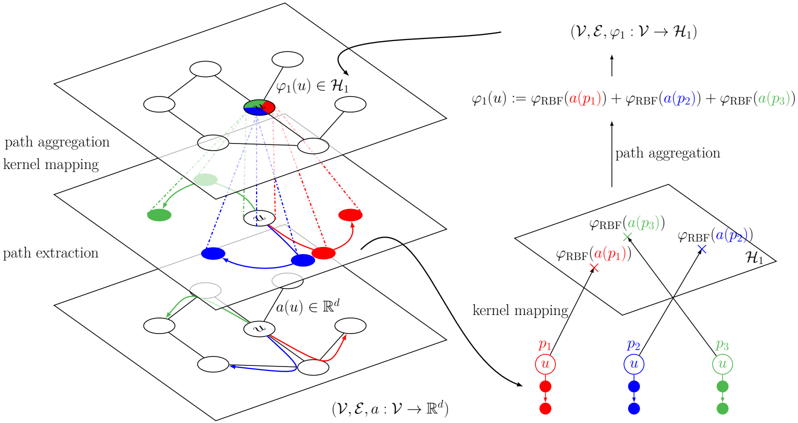
- We define the one-layer GCKN as the relaxed path kernel mapping

$$\varphi_1(u) := \sum_{p \in \mathcal{P}_k(G,u)} e^{-\frac{\alpha_1}{2}\|a(p)-\cdot\|^2} = \sum_{p \in \mathcal{P}_k(G,u)} \varphi_{\mathrm{RBF}}(a(p)) \in \mathcal{H}_1.$$

- This formula can be divided into **3 steps**:
  - path extraction: enumerating all $\mathcal{P}_k(G, u)$;
  - kernel mapping: evaluating Gaussian embedding φ_{RBF} of path features;
  - path aggregation: aggregating the path embeddings.
- We obtain a new graph with the same topology but different features

$$(\mathcal{V}, \mathcal{E}, a) \xrightarrow{\varphi_{\mathrm{path}}} (\mathcal{V}, \mathcal{E}, \varphi_1).$$

Construction of one-layer GCKN



From one-layer to multilayer GCKN

- We can repeat applying φ_{path} to the new graph

$$(\mathcal{V}, \mathcal{E}, a) \xrightarrow{\varphi_{\text{path}}} (\mathcal{V}, \mathcal{E}, \varphi_1) \xrightarrow{\varphi_{\text{path}}} (\mathcal{V}, \mathcal{E}, \varphi_2) \xrightarrow{\varphi_{\text{path}}} \dots \xrightarrow{\varphi_{\text{path}}} (\mathcal{V}, \mathcal{E}, \varphi_j).$$

- Final graph representation at layer j, $\Phi(G) = \sum_{u \in \mathcal{V}} \varphi_j(u)$.

From one-layer to multilayer GCKN

- We can repeat applying φ_{path} to the new graph

$$(\mathcal{V}, \mathcal{E}, a) \xrightarrow{\varphi_{\text{path}}} (\mathcal{V}, \mathcal{E}, \varphi_1) \xrightarrow{\varphi_{\text{path}}} (\mathcal{V}, \mathcal{E}, \varphi_2) \xrightarrow{\varphi_{\text{path}}} \ldots \xrightarrow{\varphi_{\text{path}}} (\mathcal{V}, \mathcal{E}, \varphi_j).$$

- Final graph representation at layer j, $\Phi(G) = \sum_{u \in \mathcal{V}} \varphi_j(u)$.
- Why is the multilayer model interesting ?
  - applying φ_{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:

$$\text{Span}(\varphi_{\text{RBF}}(z_1), \ldots, \varphi_{\text{RBF}}(z_q)) \text{ parametrized by } Z = \{z_1, \ldots, z_q\},$$

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:

  $$\text{Span}(\varphi_{\text{RBF}}(z_1), \ldots, \varphi_{\text{RBF}}(z_q)) \text{ parametrized by } Z = \{z_1, \ldots, z_q\},$$

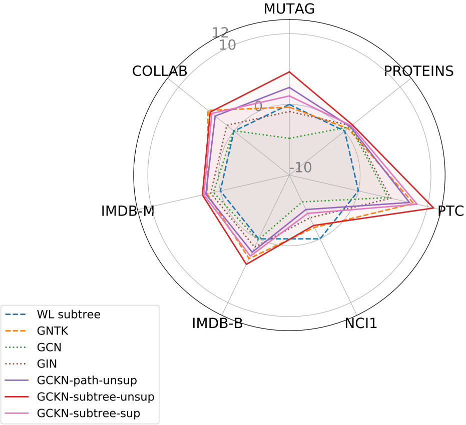
  where $z_j \in \mathbb{R}^{dk}$ can be interpreted as path features.
- The parameters Z can be learned by
  - (unsupervised) K-means on the set of path features;
  - (supervised) end-to-end learning with back-propagation.

(Chen et al., 2019a,b; Williams and Seeger, 2001)

Comparison of GCKN and GNN

GCKN	vs.	GNN
$f_{\mathsf{GCKN}}(G) = \sum_{u \in G} \psi_k(u)$ $\psi_k(u) = \sum_{p \in \mathcal{P}_k(G,u)} \kappa(Z^\top Z)^{-\frac{1}{2}} \kappa(Z^\top \psi_{k-1}(p))$		$f_{\mathsf{GNN}}(G) = \sum_{u \in G} f_k(u)$ $f_k(u) = \sum_{v \in \mathcal{N}(u)} \mathrm{ReLU}(Z^\top f_{k-1}(v))$
local path aggregation		neighborhood aggregation
projection in a known RKHS		?
supervised and unsupervised		supervised

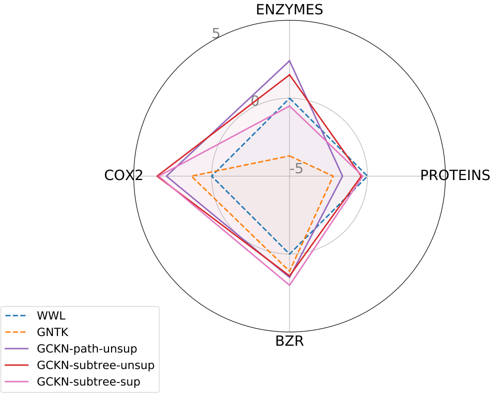
Experiments on graphs with discrete attributes



- 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

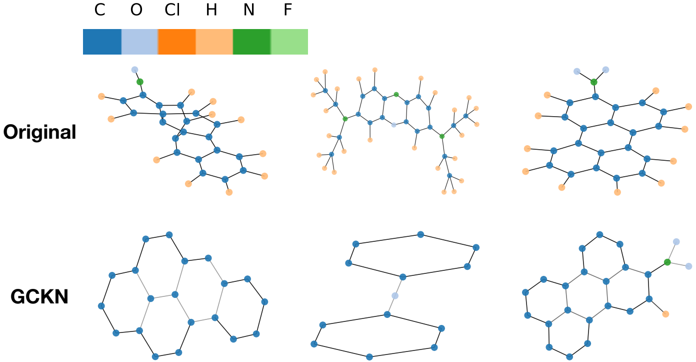


- Accuracy improvement with respect to the WWL kernel.
- Results similar to discrete case.
- Path features seem presumably predictive enough.

(Du et al., 2019; Togninalli et al., 2019)

Model interpretation for Mutagenicity prediction

- Idea: find the minimal connected component that preserves the prediction.

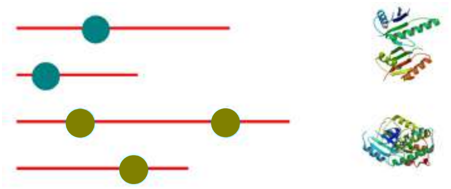


(Ying et al., 2019)

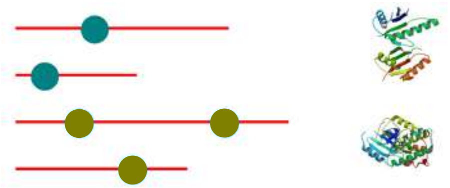
Outline

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

Sequence modeling as a supervised learning problem



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 $f : \mathcal{X} \to \mathbb{R}$

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

- 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.

$$K(x, x') = \sum_{u \in \mathcal{A}^k} \delta_u(x)\delta_u(x') \qquad ,$$

where u is a k-mer over an alphabet $\mathcal{A}$ and $\delta_u(x)$ can be:

- the number of occurrences of u in $\mathbf{x}$: **spectrum kernel** (Leslie et al., 2002);
- the number of occurrences of u in $\mathbf{x}$ up to m mismatches: **mismatch kernel** (Leslie and Kuang, 2004);
- the number of occurrences of u in $\mathbf{x}$ 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.

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 $\mathbf{x}$: **spectrum kernel** (Leslie et al., 2002);
- the number of occurrences of u in $\mathbf{x}$ up to m mismatches: **mismatch kernel** (Leslie and Kuang, 2004);
- the number of occurrences of u in $\mathbf{x}$ 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.

Convolutional kernel networks for sequence modeling

Define a continuous relaxation of the mismatch kernel (Chen et al., 2019a; Morrow et al., 2017)

$$K_{\mathsf{CKN}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{|\mathbf{x}|-k+1} \sum_{j=1}^{|\mathbf{x}'|-k+1} K_0(\underbrace{\mathbf{x}_{[i:i+k]}}_{\text{one k-mer}}, \mathbf{x}'_{[j:j+k]}).$$

- Use one-hot encoding

$$\mathbf{x}_{[i:i+5]} := \mathrm{TTGAG} \mapsto \begin{array}{c} \mathrm{A} \\ \mathrm{T} \\ \mathrm{C} \\ \mathrm{G} \end{array} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

- K_0 is a Gaussian kernel over **one-hot** representations of k-mers (in $\mathbb{R}^{k \times d}$).

Convolutional kernel networks for sequence modeling

Define a continuous relaxation of the mismatch kernel (Chen et al., 2019a; Morrow et al., 2017)

$$K_{\mathsf{CKN}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{|\mathbf{x}|-k+1} \sum_{j=1}^{|\mathbf{x}'|-k+1} \langle \varphi_0(\underbrace{\mathbf{x}_{[i:i+k]}}_{\text{one k-mer}}), \varphi_0(\mathbf{x}'_{[j:j+k]}) \rangle.$$

- Use one-hot encoding

$$\mathbf{x}_{[i:i+5]} := \mathrm{TTGAG} \mapsto \begin{array}{c} \mathrm{A} \\ \mathrm{T} \\ \mathrm{C} \\ \mathrm{G} \end{array} \left[\begin{array}{ccccc} 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array} \right].$$

- K_0 is a Gaussian kernel over **one-hot** representations of k-mers (in $\mathbb{R}^{k \times d}$).

Convolutional kernel networks for sequence modeling

Define a continuous relaxation of the mismatch kernel (Chen et al., 2019a; Morrow et al., 2017)

$$K_{\mathsf{CKN}}(\mathbf{x},\mathbf{x}') = \left\langle \underbrace{\sum_{i=1}^{|\mathbf{x}|-k+1} \varphi_0(\mathbf{x}_{[i:i+k]})}_{\Phi(x)}, \underbrace{\sum_{j=1}^{|\mathbf{x}'|-k+1} \varphi_0(\mathbf{x}'_{[j:j+k]})}_{\Phi(x')} \right\rangle .$$

- Use one-hot encoding

$$\mathbf{x}_{[i:i+5]} := \mathrm{TTGAG} \mapsto \begin{array}{c} \mathrm{A} \\ \mathrm{T} \\ \mathrm{C} \\ \mathrm{G} \end{array} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

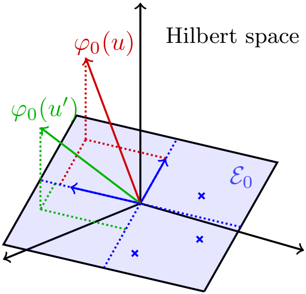
- K_0 is a Gaussian kernel over **one-hot** representations of k-mers (in $\mathbb{R}^{k\times d}$).

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:

$$\mathcal{E}_0 = \text{Span}(\varphi_0(z_1), \ldots, \varphi_0(z_q)) \text{ parametrized by } Z = \{z_1, \ldots, z_q\}.$$



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)

$$K_0(u,u') = \langle \varphi_0(u), \varphi_0(u') \rangle_{\mathcal{H}_0} \approx \langle \psi_0(u), \psi_0(u') \rangle_{\mathbb{R}^q}.$$

- **Nyström** provides a **finite-dimensional** approximation $\psi_0(u)$ in $\mathbb{R}^q$ by orthogonally projecting $\varphi_0(u)$ onto some finite-dimensional subspace:

  $$\mathcal{E}_0 = \text{Span}(\varphi_0(z_1), \dots, \varphi_0(z_q)) \text{ parametrized by } Z = \{z_1, \dots, z_q\}.$$

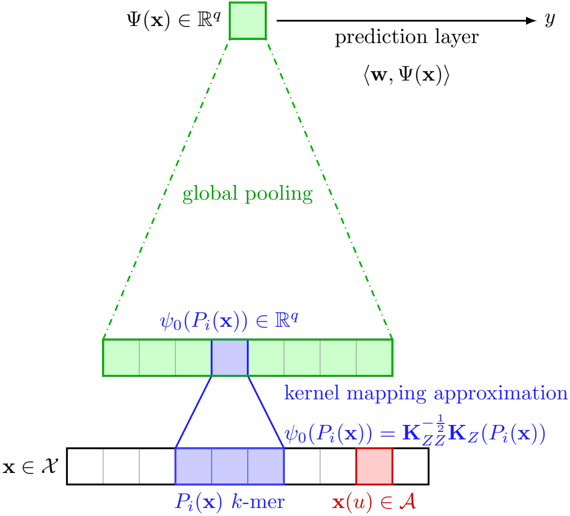
- Case of **dot-product kernels** $K_0(u,u') = \kappa(\langle u, u' \rangle)$:

  $$\psi_0(u) = \kappa(Z^\top Z)^{-1/2} \kappa(Z^\top u).$$

  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



$\Psi(\mathbf{x}) \in \mathbb{R}^q$

y

prediction layer

$\langle \mathbf{w}, \Psi(\mathbf{x}) \rangle$

global pooling

$\psi_0(P_i(\mathbf{x})) \in \mathbb{R}^q$

kernel mapping approximation

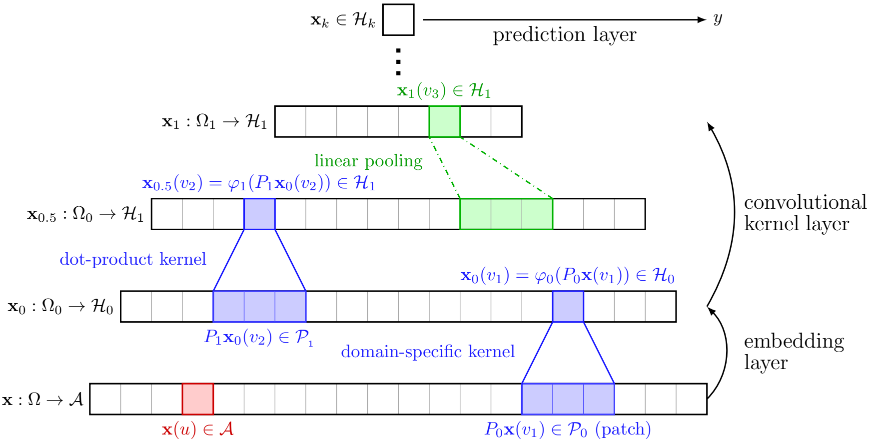
$\psi_0(P_i(\mathbf{x})) = \mathbf{K}_{ZZ}^{-\frac{1}{2}} \mathbf{K}_Z(P_i(\mathbf{x}))$

$\mathbf{x} \in \mathcal{X}$

$P_i(\mathbf{x})$ k-mer

$\mathbf{x}(u) \in \mathcal{A}$

Multilayer CKN for sequence modeling



From k-mers to gapped k-mers

k-mers with gaps

- For a sequence $\mathbf{x} = x_1 \dots x_n \in \mathcal{X}$ of length n and a sequence of ordered indices $\imath = (i_1, \dots, i_k)$ in $\mathbf{I}(k, n)$, we define a k-substring as:

$$\mathbf{x}_{[\imath]} = x_{i_1} x_{i_2} \dots x_{i_k}.$$

- We introduce the quantity

$$\text{gaps}(\imath) = \text{number of gaps in index sequence.}$$

- Example: $\mathbf{x} = \text{ABRACADABRA}$

$$\imath = (4, 5, 8, 9, 11) \quad \mathbf{x}_{[\imath]} = \text{RADAR} \quad \text{gaps}(\imath) = 3.$$

Recurrent kernel networks

Comparing all the k-mers between a pair of sequences (single layer models)

$$K_{\mathrm{CKN}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{|\mathbf{x}|-k+1} \sum_{j=1}^{|\mathbf{x}'|-k+1} K_0\left(\mathbf{x}_{[i:i+k]}, \mathbf{x}'_{[j:j+k]}\right).$$

- The kernel mapping is $\Phi(\mathbf{x}) = \sum_{i=1}^{|x|-k+1} \varphi_0(\mathbf{x}_{[i:i+k]})$.

Recurrent kernel networks

Comparing all the gapped k-mers between a pair of sequences (single layer models)

$$K_{\mathrm{RKN}}(\mathbf{x},\mathbf{x}') = \sum_{\imath \in \mathbf{I}(k,|\mathbf{x}|)} \sum_{\jmath \in \mathbf{I}(k,|\mathbf{x}'|)} \lambda^{\mathrm{gaps}(\imath)} \lambda^{\mathrm{gaps}(\jmath)} K_0\left(\mathbf{x}_{[\imath]}, \mathbf{x}'_{[\jmath]}\right).$$

- The kernel mapping is $\Phi(\mathbf{x}) = \sum_{\imath \in \mathbf{I}(k,|\mathbf{x}|)} \lambda^{\mathrm{gaps}(\imath)} \varphi_0(\mathbf{x}_{[\imath]})$.
- 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

$$\Psi(\mathbf{x}) = \sum_{\imath \in \mathbf{I}(k,t)} \lambda^{\mathrm{gaps}(\imath)} \psi_0(\mathbf{x}_{[\imath]}) \in \mathbb{R}^q,$$

where, as usual with a dot-product kernel,
$\psi_0(\mathbf{x}_{[\imath]}) = \kappa(Z^\top Z)^{-1/2}\kappa(Z^\top \mathbf{x}_{[\imath]})$.

- 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 $\mathbf{x}_{[\imath]}$, weighted by the corresponding penalization $\lambda^{\mathrm{gaps}(\imath)}$.

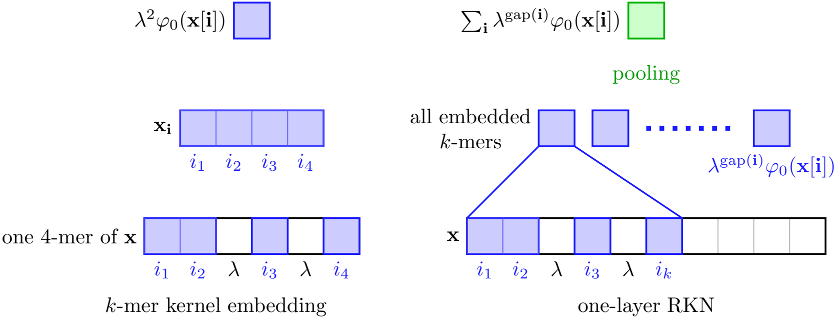


Figure: Example of K_{RKN} for $k = 4$

Results

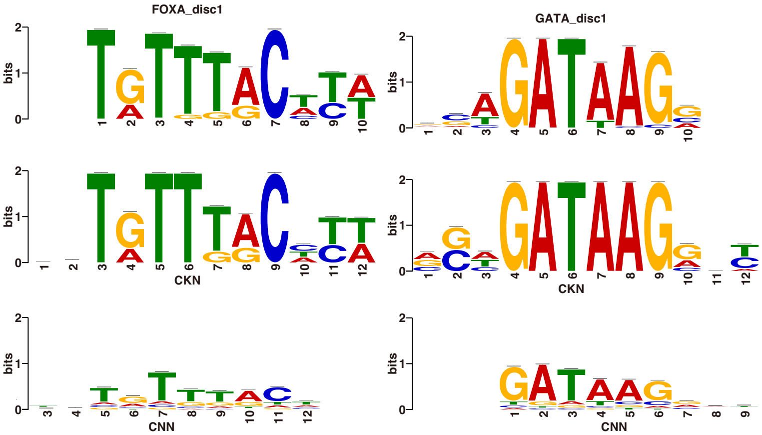
Protein fold classification on SCOP 2.06 (Hou et al., 2017) (using more informative sequence features including PSSM, secondary structure and solvent accessibility)

Method	♯Params	Accuracy		Level-stratified accuracy (top1/top5)		
		top 1	top 5	family	superfamily	fold
PSI-BLAST	-	84.53	86.48	82.20/84.50	86.90/88.40	18.90/35.100
DeepSF	920k	73.00	90.25	75.87/91.77	72.23/90.08	51.35/67.57
CKN (128 filters)	211k	76.30	92.17	83.30/94.22	74.03/91.83	43.78/67.03
CKN (512 filters)	843k	84.11	94.29	**90.24**/**95.77**	82.33/94.20	45.41/69.19
RKN (128 filters)	211k	77.82	92.89	76.91/93.13	78.56/92.98	60.54/83.78
RKN (512 filters)	843k	**85.29**	**94.95**	84.31/94.80	**85.99**/**95.22**	**71.35**/**84.86**

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)

Method	pooling	one-hot		BLOSUM62	
		auROC	auROC50	auROC	auROC50
SVM-pairwise		0.724	0.359		
Mismatch		0.814	0.467		
LA-kernel		–	–	0.834	0.504
LSTM		0.830	0.566	–	–
CKN		0.837	0.572	0.866	0.621
RKN	mean	0.829	0.541	0.840	0.571
RKN	max	**0.844**	**0.587**	**0.871**	**0.629**
RKN (unsup)	mean	0.805	0.504	0.833	0.570

(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 I

B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

F. Anselmi, L. Rosasco, C. Tan, and T. Poggio. Deep convolutional networks are hierarchical kernel machines. *arXiv preprint arXiv:1508.01084*, 2015.

N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68:337 – 404, 1950. URL `http://www.jstor.org/stable/1990404`.

F. R. Bach. Sharp analysis of low-rank kernel matrix approximations. In *COLT*, volume 30, pages 185–209, 2013.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 6, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/1015330.1015424.

P. Bartlett, M. Jordan, and J. McAuliffe. Convexity, classification and risk bounds. Technical Report 638, UC Berkeley Statistics, 2003.

P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1706.08498*, 2017.

C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic analysis on semigroups*. Springer-Verlag, New-York, 1984.

References II

A. Bietti and J. Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *J. Mach. Learn. Res.*, 2019a.

A. Bietti and J. Mairal. On the inductive bias of neural tangent kernels. In *Adv. NeurIPS*, 2019b.

L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 74–81, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2278-5. doi: http://dx.doi.org/10.1109/ICDM.2005.132.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: probability and statistics*, 9:323–375, 2005.

J. V. Bouvrie, L. Rosasco, and T. Poggio. On invariance in hierarchical models. In *Adv. NIPS*, 2009.

D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE T. Pattern. Anal.*, 35 (8):1872–1886, 2013. doi: 10.1109/TPAMI.2012.230. URL `http://dx.doi.org/10.1109/TPAMI.2012.230`.

References III

A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007. doi: 10.1007/s10208-006-0196-8. URL `https://doi.org/10.1007/s10208-006-0196-8`.

D. Chen, L. Jacob, and J. Mairal. Biological sequence modeling with convolutional kernel networks. *Bioinformatics*, 35(18):3294–3302, 2019a.

D. Chen, L. Jacob, and J. Mairal. Recurrent kernel networks. In *Adv. NeurIPS*, 2019b.

L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming. 2019.

Y. Cho and L. K. Saul. Kernel methods for deep learning. In *Adv. NIPS*, 2009a.

Y. Cho and L. K. Saul. Kernel methods for deep learning. 2009b.

T. Cohen and M. Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, 2016.

M. Cuturi and J.-P. Vert. The context-tree kernel for strings. *Neural Network.*, 18(4): 1111–1123, 2005. doi: 10.1016/j.neunet.2005.07.010. URL `http://dx.doi.org/10.1016/j.neunet.2005.07.010`.

M. Cuturi, K. Fukumizu, and J.-P. Vert. Semigroup kernels on measures. *J. Mach. Learn. Res.*, 6:1169–1198, 2005. URL `http://jmlr.csail.mit.edu/papers/v6/cuturi05a.html`.

A. Damianou and N. Lawrence. Deep Gaussian processes. In *Proc. AISTATS*, 2013.

L. Debnath and P. Mikusiński. *Introduction to Hilbert Spaces with Applications*. Elsevier, 3 edition, 2005.

References IV

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1646–1654, 2014.

L. Dicker, D. Foster, and D. Hsu. Kernel methods and regularization techniques for nonparametric regression: Minimax optimality and adaptation. Technical report, Rutgers University, 2015. URL `http://www.stat.rutgers.edu/home/ldicker/papers/kernels.pdf`.

S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Adv. NeurIPS*, 2019.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *J. Mach. Learn. Res.*, 2:243–264, 2001.

T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: hardness results and efficient alternatives. In B. Schölkopf and M. Warmuth, editors, *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory and the Seventh Annual Workshop on Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143, Heidelberg, 2003. Springer. doi: 10.1007/b12006. URL `http://dx.doi.org/10.1007/b12006`.

Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, pages 1–8. IEEE Computer Society, 2007. doi: 10.1109/CVPR.2007.383049. URL `http://dx.doi.org/10.1109/CVPR.2007.383049`.

References V

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz, 1999.

C. Helma, T. Cramer, S. Kramer, and L. De Raedt. Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. *J. Chem. Inf. Comput. Sci.*, 44(4): 1402–11, 2004. doi: 10.1021/ci034254q. URL `http://dx.doi.org/10.1021/ci034254q`.

S. Hochreiter, M. Heusel, and K. Obermayer. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736, 2007.

J. Hou, B. Adhikari, and J. Cheng. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 12 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx780. URL `https://doi.org/10.1093/bioinformatics/btx780`.

T. Jaakkola, M. Diekhans, and D. Haussler. A Discriminative Framework for Detecting Remote Protein Homologies. *J. Comput. Biol.*, 7(1,2):95–114, 2000. URL `http://www.cse.ucsc.edu/research/compbio/discriminative/Jaakola2-1998.ps`.

T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proc. of Tenth Conference on Advances in Neural Information Processing Systems*, 1999. URL `http://www.cse.ucsc.edu/research/ml/papers/Jaakola.ps`.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. 2018.

References VI

Y. Jiao and J.-P. Vert. The Kendall and Mallows kernels for permutations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. doi: 10.1109/TPAMI.2017.2719680. URL `http://dx.doi.org/10.1109/TPAMI.2017.2719680`.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In T. Faucett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328, New York, NY, USA, 2003. AAAI Press.

T. Kin, K. Tsuda, and K. Asai. Marginalized kernels for RNA sequence data analysis. In R. Lathtop, K. Nakai, S. Miyano, T. Takagi, and M. Kanehisa, editors, *Genome Informatics 2002*, pages 112–122. Universal Academic Press, 2002. URL `http://www.jsbi.org/journal/GIW02/GIW02F012.html`.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proc. ICLR*, 2017.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

N. M. Kriege, M. Neumann, C. Morris, K. Kersting, and P. Mutzel. A unifying view of explicit and implicit feature maps of graph kernels. *Data Mining and Knowledge Discovery*, 33(6): 1505–1547, 2019.

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

References VII

G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004a. URL `http://www.jmlr.org/papers/v5/lanckriet04a.html`.

G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004b. doi: 10.1093/bioinformatics/bth294. URL `http://bioinformatics.oupjournals.org/cgi/content/abstract/20/16/2626`.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL `http://dx.doi.org/10.1109/5.726791`.

T. Lei, W. Jin, R. Barzilay, and T. Jaakkola. Deriving neural architectures from sequence and graph kernels. In *Proc. ICML*, 2017.

C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *J. Mach. Learn. Res.*, 5:1435–1455, 2004.

C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: a string kernel for SVM protein classification. In R. B. Altman, A. K. Dunker, L. Hunter, K. Lauerdale, and T. E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing 2002*, pages 564–575, Singapore, 2002. World Scientific.

References VIII

C. Leslie, E. Eskin, J. Weston, and W. Noble. Mismatch String Kernels for SVM Protein Classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, 2003. URL `http://www.cs.columbia.edu/~cleslie/papers/mismatch-short.pdf`.

L. Liao and W. Noble. Combining Pairwise Sequence Similarity and Support Vector Machines for Detecting Remote Protein Evolutionary and Structural Relationships. *J. Comput. Biol.*, 10(6):857–868, 2003. URL `http://www.liebertonline.com/doi/abs/10.1089/106652703322756113`.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, 2002. URL `http://www.ai.mit.edu/projects/jmlr/papers/volume2/lodhi02a/abstract.html`.

B. Logan, P. Moreno, B. Suzek, Z. Weng, and S. Kasif. A Study of Remote Homology Detection. Technical Report CRL 2001/05, Compaq Cambridge Research laboratory, June 2001.

P. Mahé and J. P. Vert. Graph kernels based on tree patterns for molecules. *Mach. Learn.*, 75 (1):3–35, 2009. doi: 10.1007/s10994-008-5086-2. URL `http://dx.doi.org/10.1007/s10994-008-5086-2`.

References IX

P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*, pages 552–559. ACM Press, 2004.

P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *J. Chem. Inf. Model.*, 45(4):939–51, 2005. doi: 10.1021/ci050039t. URL `http://dx.doi.org/10.1021/ci050039t`.

J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

J. Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 1399–1407, 2016.

S. Mallat. Group invariant scattering. *Comm. Pure Appl. Math.*, 65(10):1331–1398, 2012. doi: 10.1002/cpa.21413. URL `http://dx.doi.org/10.1002/cpa.21413`.

C. Micchelli and M. Pontil. Learning the kernel function via regularization. *J. Mach. Learn. Res.*, 6:1099–1125, 2005. URL `http://jmlr.org/papers/v6/micchelli05a.html`.

G. Montavon, M. L. Braun, and K.-R. Müller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12(Sep):2563–2581, 2011.

A. Morrow, V. Shankar, D. Petersohn, A. Joseph, B. Recht, and N. Yosef. Convolutional kitchen sinks for transcription factor binding site prediction. *arXiv preprint arXiv:1706.00125*, 2017.

References X

C. Müller. *Analysis of spherical symmetries in Euclidean spaces*, volume 129 of *Applied Mathematical Sciences*. Springer, 1998.

R. M. Neal. *Bayesian learning for neural networks*. Springer, 1996.

Y. Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer Academic Publishers, 2004.

B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. In *Proc. ICLR*, 2018.

A. Nicholls. Oechem, version 1.3.4, openeye scientific software. website, 2005.

M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronin, and C. Schmid. Local convolutional features with unsupervised training for image retrieval. In *Proc. CVPR*, 2015.

F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Adv. NIPS*, 2007.

A. Raj, A. Kumar, Y. Mroueh, P. T. Fletcher, and B. Scholkopf. Local group invariant representations via orbit embeddings. *preprint arXiv:1612.01988*, 2016.

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *J. Mach. Learn. Res.*, 9:2491–2521, 2008. URL `http://jmlr.org/papers/v9/rakotomamonjy08a.html`.

J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In T. Washio and L. De Raedt, editors, *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74, 2003.

References XI

F. Rapaport, A. Zynoviev, M. Dutreix, E. Barillot, and J.-P. Vert. Classification of microarray data using gene networks. *BMC Bioinformatics*, 8:35, 2007. doi: 10.1186/1471-2105-8-35. URL `http://dx.doi.org/10.1186/1471-2105-8-35`.

H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004. URL `http://bioinformatics.oupjournals.org/cgi/content/abstract/20/11/1682`.

M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2016.

I. Schoenberg. Positive definite functions on spheres. *Duke Math. J.*, 1942.

B. Scholkopf. *Support Vector Learning*. PhD thesis, Technischen Universität Berlin, 1997.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002. URL `http://www.learning-with-kernels.org`.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT Press, The MIT Press, Cambridge, Massachussetts, 2004.

M. Seeger. Covariance Kernels from Bayesian Generative Models. In *Adv. Neural Inform. Process. Syst.*, volume 14, pages 905–912, 2002.

References XII

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 2015.

V. Shankar, A. Fang, W. Guo, S. Fridovich-Keil, L. Schmidt, J. Ragan-Kelley, and B. Recht. Neural kernels without tangents. *preprint arXiv:2003.02237*, 2020.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004a.

J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004b.

N. Shervashidze and K. M. Borgwardt. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems*, pages 1660–1668, 2009.

N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 488–495, Clearwater Beach, Florida USA, 2009. Society for Artificial Intelligence and Statistics.

N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *The Journal of Machine Learning Research*, 12: 2539–2561, 2011.

References XIII

L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2013.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.

T. Smith and M. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

A. J. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 911–918, San Fransisco, CA, USA, 2000. Morgan Kaufman.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

M. Togninalli, E. Ghisu, F. Llinares-López, B. Rieck, and K. Borgwardt. Wasserstein Weisfeiler-Lehman graph kernels. In *Adv. NeurIPS*, 2019.

K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414, 2002a. doi: 10.1162/08997660260293274. URL `http://dx.doi.org/10.1162/08997660260293274`.

K. Tsuda, T. Kin, and K. Asai. Marginalized Kernels for Biological Sequences. *Bioinformatics*, 18:S268–S275, 2002b.

A. B. Tsybakov. *Introduction à l'estimation non-paramétrique*. Springer, 2004.

References XIV

V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0387945598. URL `http://portal.acm.org/citation.cfm?id=211359`.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New-York, 1998.

J.-P. Vert, H. Saigo, and T. Akutsu. Local alignment kernels for biological sequences. In B. Schölkopf, K. Tsuda, and J. Vert, editors, *Kernel Methods in Computational Biology*, pages 131–154. MIT Press, The MIT Press, Cambridge, Massachussetts, 2004a.

J.-P. Vert, H. Saigo, and T. Akutsu. Convolution and local alignment kernels. *Kernel methods in computational biology*, pages 131–154, 2004b.

J.-P. Vert, R. Thurman, and W. S. Noble. Kernels for gene regulatory regions. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Adv. Neural. Inform. Process Syst.*, volume 18, pages 1401–1408, Cambridge, MA, 2006. MIT Press.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

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.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Adv. NIPS*, 2001.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

References XV

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *Proc. ICLR*, 2019.

Y. Yamanishi, J.-P. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20:i363–i370, 2004. URL `http://bioinformatics.oupjournals.org/cgi/reprint/19/suppl_1/i323`.

Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Adv. NeurIPS*, 2019.

Y. Zhang, J. D. Lee, and M. I. Jordan. ℓ_1-regularized neural networks are improperly learnable in polynomial time. In *International Conference on Machine Learning (ICML)*, 2016.

Y. Zhang, P. Liang, and M. J. Wainwright. Convexified convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2017.