

Kernel Methods in Machine Learning

Graduate Machine Learning Seminar, NKUA

Maritina Marouli

Department of Mathematics

March 20, 2026

- Motivation
- Mercer Kernel and Spaces in Functional Analysis
- Feature Maps
- Properties of kernels
- Examples of kernel functions
- Kernel Centering and Normalization
- Kernelization Techniques
- Use of kernels in Machine Learning

Many real-world datasets are not linearly separable in their original feature space, limiting simple models. Kernel methods address this by implicitly mapping data to high-dimensional spaces using kernel functions, allowing complex patterns to be captured efficiently. Through the kernel trick and the Representer theorem, solutions can be expressed in terms of a few computable coefficients, making kernel methods both powerful and computationally practical for tasks like classification and dimensionality reduction.

- **Mercel Kernel Function:** the function $k : X^2 \rightarrow \mathbb{R}$ is a Mercer Kernel Function where it is symmetric ($k(x, y) = k(y, x)$) and its corresponding kernel matrix $K(i, j) = k(x_i, x_j)$, $i, j \in \{1, 2, \dots, n\}$ is positive semi-definite. Then K is called a Mercer or Gram matrix.

- **Hilbert Space:** a Hilbert space $\mathcal{H}$ is an inner product space that is a complete metric space with respect to the norm induced by its inner product.
- **$\mathcal{L}_p$ Space:** for $p > 0$ we define the p-norm: $\|f\|_p = (\int |f(x)|^p dx)^{\frac{1}{p}}$. Then the $\mathcal{L}_p$ space is defined as the set of functions with bounded $\mathcal{L}_p$ norm: $\mathcal{L}_p(a, b) = \{f : [a, b] \rightarrow \mathbb{R} : \|f\|_p < \infty\}$.

Mercel Kernel and Spaces in Functional Analysis

Consider the kernel $k(x, y)$, which is a function of two variables. Suppose for n points we fix one of the variables, let's say y , to have $k(x_1, y), k(x_2, y), \dots, k(x_n, y)$. These are all functions of the variable y .

RKHS is a function space which is the set of all the possible linear combinations of these functions: $\mathcal{H} := \{f(\cdot) = \sum_{i=1}^n a_i k(x_i, \cdot)\}$. The bases of this space are kernels and every function of $\mathcal{H}$ can be written as their linear combination.

Special property: if f is a function that belongs in the RKHS, it has the **reproductive property**, its value at any point is "reproduced" by the inner product with the kernel: $f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}$.

- **Reproducing Kernel Hilbert Space (RKHS):** a Hilbert space $\mathcal{H}$ of functions $f : X \rightarrow \mathbb{R}$ with a reproducing kernel $k : X^2 \rightarrow \mathbb{R}$ where $k(x, \cdot) \in \mathcal{H}$ and $f(x) = \langle k(x, \cdot), f \rangle$.
- **Special Case:** $\langle k(x, \cdot), k(y, \cdot) \rangle_{\mathcal{H}} = k(x, y)$.
- **Inner product in RKHS:**
$$\langle f, g \rangle_k = \langle \sum a_i k(x_i, \cdot), \sum b_j k(\cdot, y_j) \rangle_k = \sum \sum a_i b_j k(x_i, y_j)$$

We get the norm: $\|f\|_k = \sqrt{\langle f, f \rangle_k}$

Mercel Kernel and Spaces in Functional Analysis

The functions in the Hilbert space need to become "calculable". We use feature maps to translate them into vectors (we will see later how). Then, the vectors represent the position of the arguments after being transformed into the high dimensional feature space.

Concept	Function World	Vector World
point x	input value	position vector $\phi(x)$
function f	rule $x \mapsto y$	direction vector w
evaluation of $f(x)$	plugging x into a formula	dot product $w \cdot \phi(x)$
kernel(x,y)	similarity measure	dot product $\phi(x) \cdot \phi(y)$

EXAMPLE

- **The kernel:** $k(x, y) = xy + 1$. If $\phi(x) = [x \ 1]^T$, then $\phi(x)^T \phi(y) = \langle \phi(x), \phi(y) \rangle = [x \ 1][y \ 1]^T = xy + 1 \Rightarrow k(x, y) = \langle \phi(x), \phi(y) \rangle$.
- **The Hilbert Space:** all of the functions that can be written as a linear combination of the kernel calculated in different points.
- **The function:** $f(x) = 2x + 5$.
- **The inner product:** $\langle f, k(x, \cdot) \rangle_{\mathcal{H}} = [2 \ 5][x \ 1]^T = 2x + 5 = f(x)$

Mercel Kernel and Spaces in Functional Analysis

- Given a kernel, the corresponding RKHS is unique.
- Given a RKHS, the corresponding kernel is unique.

(up to isometric isomorphisms)

The Representation Theorem

For a set of data $X = \{x_i\}_{i=1}^n$ consider a RKHS $\mathcal{H}$ of functions $f : X \rightarrow \mathbb{R}$ with kernel function k . For any function $l : \mathbb{R}^2 \rightarrow \mathbb{R}$ (loss) consider the optimization problem:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^n l(f(x_i), y_i) + \eta \mathcal{W}(\|f\|_k)$$

where $\eta \geq 0$ regularization parameter, $\mathcal{W}(\|f\|_k)$ a penalty term (ex. $\|f\|_k^2$).

The optimal solution f^* always takes the form: $f^* = \sum_{i=1}^n a_i k(x_i, \cdot)$ (finite form, even though the Hilber Space might be infinite dimensional).

This means the best function is just a weighted sum of the kernels centered at our training data points and we get solutions for problems such as SVMs.

Mercer's Theorem

If $k : [a, b] \times [a, b] \rightarrow \mathbb{R}$ is a continuous, symmetric positive semi-definite kernel, then:

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

where $\{\psi_i\}$, $\{\lambda_i\}$ are the eigenfunctions and eigenvalues of the kernel operator.

Feature Maps

We define a **mapping** $\phi : X \rightarrow \mathcal{H}$ to transform data from input space to feature space:

$$\phi(x) = [\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \dots]^T$$

Then $k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} = \phi(x)^T \phi(y)$: where $\{\psi_i\}$, $\{\lambda_i\}$ are the eigenfunctions and eigenvalues of the kernel operator.

Feature Maps

The kernel between two points is the inner product of pulled data points to the feature space. If we have n observations $x_1, x_2, \dots, x_n \in X$ and stack their maps we get $\Phi(x) = [\phi(x_1), \phi(x_2), \dots, \phi(x_n)]$ with dimensions $t \times n$. Then

$$\mathbb{R}^{n \times n} \ni K = \langle \Phi(x), \Phi(x) \rangle_k = \Phi(x)^T \Phi(x)$$

Instead of computing eigenfunctions we can simply calculate inner product suffices.

Feature Maps

The space in which data X exists is called **input/original** space. It is denoted by X and usually $X \subset \mathbb{R}^d$. The RKHS to which the data is pulled is called **feature** space. Data can be pulled from the input space using kernels. The relative similarity (inner product) of pulled data points is known by the kernel. However, the exact location of pulled data points to RKHS is not necessarily known.

Properties of kernels

- 1 A square Mercer kernel matrix is symmetric:
 $\langle f, g \rangle_k = \langle g, f \rangle_k \Rightarrow k(x, y) = k(y, x)$
- 2 $\langle f, f \rangle_k = 0 \Leftrightarrow f = 0$
- 3 The Mercer kernel is positive semi-definite: $K \geq 0$

Examples of kernel functions

- **Linear kernel:** $k(x, y) = x^T y$. It has the explicitly known feature map $\phi(x) = x$. The data is not pulled in a different feature space, we just calculate their inner product.
- **Gaussian (RBF):** $k(x, y) = e^{-\gamma \|x-y\|^2}$ where $\gamma = \frac{1}{\sigma^2}$ and σ^2 is the variance of the kernel (changes the punishment we pose to their distance).
- **Laplacian:** $k(x, y) = e^{-\gamma \|x-y\|_1}$ where $\gamma = \frac{1}{\sigma^2}$.
- **Sigmoid:** $k(x, y) = \tanh(\gamma x^T y + c)$ where $\gamma > 0$ slope, c intercept
- **Polynomial:** $k(x, y) = (\gamma x^T y + c)^d$ where $\gamma > 0$ slope, c intercept.
- **Chi-squared kernel:** $k(x, y) = e^{-\gamma \sum \frac{(x_i - y_i)^2}{x_i + y_i}}$ where $\gamma > 0$ parameter.

Examples of kernel functions

Important classes of kernels

- **Bounded:** A kernel function k is **bounded** if $\sup k(x, y) < \infty$.
- **Universal:** If we denote by $C(X)$ the space of all continuous functions on space X , a continuous kernel k on a compact metric space X is called **universal** if the RKHS $\mathcal{H}$ with kernel function k is dense in $C(X)$ (for every $g \in C(X) \forall \epsilon > 0 \exists f \in \mathcal{H} : \|f - g\|_\infty \leq \epsilon$).
We can approximate any function using a universal kernel.
- **Stationary:** A kernel k is **stationary** if it's a positive definite function of the form $k(x, y) = k(\|x - y\|)$ where $\|\cdot\|$ is some norm defined on the input space. The inner product is a non-stationary kernel. **A stationary kernel views the world uniformly: the notion of similarity is the same regardless of where you are.**

Kernel Centering and Normalization

In some cases, there is a need to center the pulled data in the feature space. For this, the kernel matrix should be centered in a way that the mean of pulled dataset becomes zero.

Also, kernel value can be large if the pulled vectors to the feature map have large length. In practical computation and optimization, it is sometimes required to normalize the kernel matrix.

Kernel Centering and Normalization

Centering: Assume we have some training data $X = (x_1, \dots, x_n) \in \mathbb{R}^{d \times n}$. Consider the kernel matrix of the training data $K = \Phi(X)^T \Phi(X) : K(i, j) = \phi(x_i)^T \phi(x_j) \in \mathbb{R}$. We want to center the pulled training data in the feature $\tilde{\phi}(x_i) = \phi(x_i) - \frac{1}{n} \sum_m \phi(x_m)$. Then the (i,j)-element of the kernel matrix becomes

$$\tilde{K}(i, j) = \tilde{\phi}(x_i)^T \tilde{\phi}(x_j) = K - \frac{1}{n} \mathbf{1} \mathbf{1}^T K - \frac{1}{n} K \mathbf{1} \mathbf{1}^T + \frac{1}{n^2} \mathbf{1} \mathbf{1}^T K \mathbf{1} \mathbf{1}^T =$$

$$\left(I_n - \frac{1}{n} \mathbf{1} \mathbf{1}^T \right) K \left(I_n - \frac{1}{n} \mathbf{1} \mathbf{1}^T \right) = H K H$$

This is called the **double centered kernel**, with **centering matrix** $H = I_n - \frac{1}{n} \mathbf{1} \mathbf{1}^T$.

Kernel Centering and Normalization

This equation represents the kernel matrix when the pulled training data feature space is centered. It has zero row-wise and column-wise mean. After this kernel centering we get:

$$\frac{1}{n} \sum_i \tilde{\phi}(x_i) = 0, \quad \sum_i \sum_j \tilde{K}(i, j) = 0$$

Normalization: The kernel matrix $K \in \mathbb{R}^{n \times n}$ can be normalized as

$$\frac{K(i,j)}{\sqrt{K(i,i)K(j,j)}} \mapsto K(i,j) \quad \forall (i,j) \in \{1, 2, \dots, n\}$$

. All its diagonal values are ones.

Linear algorithms obviously cannot properly handle nonlinear patterns of data. When dealing with non-linear data, if the algorithm is linear, two solutions exist to have acceptable performance:

- 1 Either the linear method should be modified to become nonlinear, or a completely new non-linear algorithm should be proposed to be able to handle non-linear data (ex: locally linear embedding).
- 2 Or the non-linear data should be modified in a way to become more linear in pattern (transformation to make them easier to process by the linear algorithm - ex: kernel PCA, kernel FDA, kernel SVM).

In machine learning and data science, **kernelization** means a slight change in algorithm formulation so that the pulled data to the RKHS are used as input of algorithm, rather than raw data.

Kernelization by Kernel Trick

In this technique we first try to formulate the algorithm optimization in a way that data always appears as an inner product of data instances: $x^T x$, $X^T x$, $x^T X$, $X^T X$, not a lonely x or X . In this way, "kernel trick" replaces $x^T x$ with $\phi(x)^T \phi(x)$.

$$x^T x \mapsto \phi(x)^T \phi(x) = k(x, x)$$

$$X^T X \mapsto \Phi(X)^T \Phi(X) = K(X, X)$$

In some algorithms data do not appear only by inner product, which is required by the kernel trick. If possible, a "dual" method for the algorithm is proposed, which only uses the inner product of data. Then the dual algorithm is kernelized using the kernel trick, ex. dual SVM.

Kernelization by Representation Theory

Some algorithms cannot be formulated in a way to have data only in inner product form, nor does their dual have this form in order to perform the kernel trick.

Consider an algorithm whose optimization variable is a vector u (or matrix U) in the input space. For kernelization, we pull it to RKHS to have $\phi(u)$. This pulled solution must lie in the span of all pulled training points as $\phi(u) = \sum_i a_i \phi(x_i) = \Phi(X)a$. The vector a contains the coefficients. We can replace u with $\phi(u)$ in the algorithm. If $\phi(x_i)^T \phi(x_i)$ or $\Phi(X)^T \Phi(X)$ appear, we can replace them with $k(x_i, x_i)$ or $K(X, X)$. This kernelizes the method.

The representer theorem tells us that the solution lies in a kernel expansion, even when no inner products are explicitly present, while the kernel trick is a way to compute such expressions efficiently when they appear.

We use the kernel trick when the algorithm depends on inner products, while the representer theorem guarantees that, under regularization, the solution can be expressed as a kernel expansion—even if this structure is not immediately visible.

The representer theorem is valuable not because it always makes problems easy to solve, but because it tells us that the solution lies in a finite-dimensional subspace defined by the data.

Types of use of kernels in machine learning

The first type of using kernels in machine learning is kernelization of algorithms using either kernel trick or representation theory.

Universal kernels are widely used in kernel SVM, as they can approximate any continuous function, making the model highly flexible for capturing non-linear patterns.

Types of use of kernels in machine learning

Manifold: a space that may be complex globally but locally looks like a simply Euclidean space (ex. sphere, globally curved, locally flat).

In Machine Learning, data often lives in a high-dimensional space, but actually lies on a low dimensional manifold. The goal is to take a curved structure and "unfold" it, into a flat space, without destroying its geometry.

We need to maximize the variance - force the data to spread out as much as possible, effectively "unfolding" the manifold, with the constrain to preserve local distances.

Types of use of kernels in machine learning

$$\|\phi(x_i) - \phi(x_j)\|_k^2 = k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j)$$

Trace of matrix: the summation of eigenvalues

Kernel matrix: measure of similarities between points

Trace of kernel: summation of the variance of the data

So we seek to maximize $tr(K)$. We get the following optimization problem:

$$\max_K tr(K)$$

$$sub \|\phi(x_i) - \phi(x_j)\|_2^2 = K(i, i) + K(j, j) - 2K(i, j) \quad \forall (i, j) \in \{1, 2, \dots, n\},$$

$$\sum_{i=1}^n \sum_{j=1}^n K(i, j) = 0, \quad K \geq 0$$

This is called the MVU (Maximum Variance Unfolding) kernel selection.

Types of use of kernels in machine learning

Kernels can also be used directly for dimensionality reduction. Consider X to be a random variable of data and Y the outcome (labels - discrete and finite for classification or continuous for regression).

SDR (Sufficient Dimensionality Reduction): family of methods which find a transformation of data to a lower dimensional space $R(x)$, that doesn't change the conditional of labels given data:

$$\mathbb{P}_{Y|X}(y|x) = \mathbb{P}_{Y|X}(y|R(x))$$

KDR (Kernel Dimensionality Reduction) is an SDR method with transformation $x \mapsto U^T x$ - linear projection onto the columns space of U . The goal is:

$$\mathbb{P}_{Y|X}(y|x) = \mathbb{P}_{Y|U,X}(y|U, x)$$

References I

Reproducing Kernel Hilbert Space, Mercer's Theorem, Eigenfunctions, Nystrom Method, and Use of Kernels in Machine Learning: " Tutorial and Survey (Ghojogh, Ghodsi, Karray, Crowley, 2021)

Data Mining and Machine Learning: Fundamental Concepts and Algorithms (Zaki, Meira Jr., 1Department of Computer Science Rensselaer Polytechnic Institute, Troy, NY, USA 2Department of Computer Science Universidade Federal de Minas Gerais, Belo Horizonte, Brazil)

Thank you for your attention!

Maritina Marouli