Lecture 6 & 7: Kernel Methods:
Fundamentals and Special Topics

Readings: Bach (Ch. 7), ESL (Ch. 5.8); code

Soon Hoe Lim

September 11 & 16, 2025

Outline

1 Kernel Methods: Introduction

2 Kernels and Feature Maps

3 Kernel Ridge Regression

4 The Representer Theorem

5 Approximations with Random Features

6 Exercises

7 Appendix: Optional Materials

Connecting the Dots: The Big Picture

We've encountered several powerful, but seemingly distinct, ideas:

- ▶ In SVMs, we saw a clever **"kernel trick"** that created complex non-linear boundaries by implicitly mapping data to a new feature space.
- ▶ In non-linear modeling, we saw that **smoothing splines** find a flexible function by minimizing a combined "loss + smoothness penalty".
- ▶ Both the ridge regression and SVC also use a "loss + regularization penalty" to control complexity.

💡 This raises some deep questions:

- ▶ Is the "kernel trick" a one-off gimmick just for SVMs, or is it a more general recipe?
- ▶ Is there a formal connection between a penalty on weights like $\|\beta\|^2$ and a penalty on a function's "wiggliness"?
- ▶ Can we build a **single, unified theory** that explains all of these?

In this lecture, we will reveal the beautiful and powerful theory of **kernel methods** that connects all these dots.

Kernel Methods: Introduction

▶ General linear basis models (see Lecture 5) work by transforming inputs $x_i \in \mathbb{R}^d$ via a set of functions $\{\phi_j\}$ (**feature maps**).

▶ They extract useful features from the input data and allow us to use a linear model on the resulting space.

▶ We will develop this idea in greater generality in the context of kernel methods.
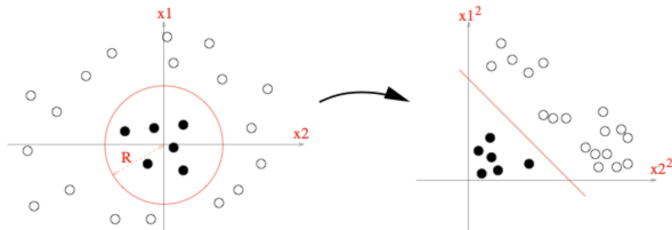
🔎 We have seen the power of using polynomial features to solve the above classification task in Lecture 5. But how about regression tasks? See blackboard.

Ridge Least Squares Revisited

We begin by revisiting the ℓ_2-regularized least squares problem (ridge regression) with respect to feature maps $\{\phi_j\}_{j=1}^M$. Recall we are given $D_N = \{(x_i, y_i)\}_{i=1}^N$.

The objective is find $\beta \in \mathbb{R}^M$ that minimizes:

$$\hat{R}_N(\beta) = \frac{1}{N} \left(\|\Phi\beta - y\|^2 + \lambda\|\beta\|^2 \right),$$

where $\Phi_{ij} = \phi_j(x_i)$ (design matrix for the features) and the regularization parameter $\lambda > 0$. Here $\Phi \in \mathbb{R}^{N \times M}$, $\mathbf{y} = (y_1, \ldots, y_N) \in \mathbb{R}^N$.

Just as in Lecture 2, the solution is given by:

$$\hat{\beta}_\lambda = (\Phi^T\Phi + \lambda\mathbf{I}_M)^{-1}\Phi^T\mathbf{y}.$$

The prediction on a new sample $x \in \mathbb{R}^d$ is:

$$\hat{f}(x) = \phi(x)^T \hat{\beta}_\lambda,$$

where $\phi(x) = (\phi_1(x), \ldots, \phi_M(x))$.

Rewriting the Solution: The Dual Form

We can rewrite the regularized least squares solution in another way.

Lemma 1: Matrix Inversion Lemma

The solution $\hat{\beta}_\lambda$ can be expressed as:

$$\hat{\beta}_\lambda = (\Phi^T \Phi + \lambda I_M)^{-1} \Phi^T \mathbf{y} = \Phi^T (\Phi \Phi^T + \lambda I_N)^{-1} \mathbf{y}.$$

Proof.
See blackboard. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

The Predictor in Dual Form

Using the dual form of $\hat{\beta}_\lambda$, the predictor function $\hat{f}(x)$ can be written as:

$$\hat{f}(x) = \phi(x)^T \hat{\beta}_\lambda = \phi(x)^T \Phi^T (\Phi\Phi^T + \lambda \mathbf{I}_N)^{-1} \mathbf{y}.$$

By defining a vector of coefficients $\alpha \in \mathbb{R}^N$ as:

$$\alpha = (\Phi\Phi^T + \lambda \mathbf{I}_N)^{-1} \mathbf{y}.$$

We can rewrite the predictor as:

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \phi(x)^T \phi(x_i).$$

What have we achieved?

▶ The $N \times N$ matrix $\Phi\Phi^T$ has components $(\Phi\Phi^T)_{ij} = \phi(x_i)^T \phi(x_j)$.

▶ The predictor formula now only depends on the feature map ϕ through the function $(x, x') \mapsto \phi(x)^T \phi(x')$.

The Kernel Trick

We can define a kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ by: $k(x, x') = \phi(x)^T \phi(x')$. The predictor can be rewritten as:

$$\hat{f}(x) = \sum_{i=1}^{N} [(K + \lambda I_N)^{-1} \mathbf{y}]_i k(x, x_i),$$

where K is the $N \times N$ Gram matrix with $K_{ij} = k(x_i, x_j)$.

▶ The predictor $\hat{f}$ only depends on the feature maps through this kernel k. Once k is known, we never need to compute the feature maps $x \mapsto \phi(x)$ directly to make a prediction.

▶ If M is very large (even infinite!), as long as we can compute the kernel k efficiently, this is a huge saving. See Exercise 6.3 for their other benefits.

▶ This is the key idea of kernel methods. The kernel trick can be applied to many cases beyond the RLS problem (e.g., logistic regression, SVM, Gaussian Processes and even unsupervised learning methods like PCA).

Kernels and Feature Maps

For any set of feature maps $\{\phi_j\}$, we can construct a kernel:

$$k(x, x') = \phi(x)^T \phi(x') = \sum_{j=1}^{M} \phi_j(x)\phi_j(x').$$

Examples:

1. **(Kernels From Feature Maps)** Let $d = 1$, $M = 2$ and $\phi(x) = (1, x)$. This is 1D simple linear regression. The kernel is:

$$k(x, x') = 1 + xx'.$$

2. **(Feature Maps From Kernels)** Consider $d = 2$ and the function $k(x, x') = (1 + x^T x')^2$. Does k correspond to a feature map?

$$\begin{aligned}
k(x, x') &= (1 + x_1 x_1' + x_2 x_2')^2 \\
&= (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2)^T (1, \sqrt{2}x_1', \ldots, x_2'^2).
\end{aligned}$$

The feature map is $\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2)$. Thus, specifying a kernel function implicitly defines a feature map here.

What is a Valid Kernel?

Can any function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel? **No.**

- A kernel must be symmetric in its arguments: $k(x, x') = k(x', x)$.
- It must satisfy $k(x, x) = \|\phi(x)\|^2 \geq 0$.
- For example, $k(x, x') = x^T x' - 1$ cannot be a kernel since $k(0, 0) = -1 < 0$.

This motivates the restrictions on what functions we consider to be valid kernels that defines feature maps.

Definition 1: Positive Definite (PD) Kernels

A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a positive definite (PD) kernel if for any collection $\{x_1, \ldots, x_N\}$ of vectors in $\mathbb{R}^d$, the Gram matrix K with elements $K_{ij} = k(x_i, x_j)$ is symmetric and positive semi-definite (PSD).

Recall: K is symmetric if $K_{ij} = K_{ji}$ for all i, j, and PSD if $c^T K c \geq 0$ for any $c \in \mathbb{R}^N$.

Mercer's Theorem

If $k(x, x') = \phi(x)^\top \phi(x')$ for some feature map ϕ, then k is automatically positive definite: it is obvious that the Gram matrix K is symmetric, and for any $c \in \mathbb{R}^N$,

$$\sum_{i,j} c_i c_j k(x_i, x_j) = \sum_{i,j} c_i c_j \langle \phi(x_i), \phi(x_j) \rangle = \left\| \sum_i c_i \phi(x_i) \right\|^2 \geq 0.$$

💡 Interestingly, the reverse is also true: by Mercer's theorem, every positive definite kernel can be written as an inner product in a (possibly infinite-dimensional) Hilbert space.

⚠️ Mercer's theorem and the rigorous foundation of kernel methods based on reproducing kernel Hilbert spaces (RKHS) are not covered in this course; see Appendix for a short introduction.

What is a Hilbert Space?

For our purposes, a Hilbert space $\mathcal{H}$ is a vector space that generalizes the familiar Euclidean space $\mathbb{R}^M$ to potentially infinite dimensions.

- ▶ It is a **vector space**, so we can add vectors and scale them by constants.
- ▶ It has an **inner product** $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ (also called a dot product). This allows us to define geometric notions like angles and lengths.
- ▶ The inner product defines a **norm** (length) for any vector $\beta \in \mathcal{H}$ as $\|\beta\|_{\mathcal{H}} = \sqrt{\langle \beta, \beta \rangle_{\mathcal{H}}}$.
- ▶ (Technically, it must also be "complete," which ensures that limits and calculus work as we expect.)

We will mainly work with models where the feature space $\mathcal{H}$ is $\mathbb{R}^M$.

- ▶ **Inner Product:** $\langle \beta, z \rangle = \beta^\top z = \sum_{j=1}^{M} \beta_j z_j$
- ▶ **Euclidean Norm:** $\|\beta\| = \sqrt{\beta^\top \beta} = \sqrt{\sum_{j=1}^{M} \beta_j^2}$

💡 The goal of kernel methods is to handle cases where M is very large or even infinite, without ever working in that space directly.

Examples of Positive Definite Kernels

1. **Linear kernel:** $k(x, x') = x^T x'$.

2. **Polynomial kernel:** $k(x, x') = (1 + x^T x')^m, m > 0$.

3. **Gaussian (RBF) kernel:** $k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right), \sigma > 0$.

4. **Laplacian kernel:** $k(x, x') = \exp(-\gamma\|x - x'\|), \gamma > 0$.

5. **Matérn Kernels**[1]**:** $k_{\nu,\ell}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|x - x'\|}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}\|x - x'\|}{\ell}\right)$, for parameters $\nu, \ell > 0$, where Γ is the Gamma function and K_{ν} is a modified Bessel function of the second kind. For $\nu = 1/2$, this simplifies to the exponential kernel. More generally, a smaller ν results in less smooth functions, and as $\nu \to \infty$, the Matérn kernel converges to the Gaussian kernel. Popular choices are $\nu = 3/2$ and $\nu = 5/2$ (see Exercise 6.1).

6. **Kernel on Sets:** For a finite set Ω, let $A, A' \subseteq \Omega$. $k(A, A') = 2^{|A \cap A'|}$ is an PD kernel. This shows kernel methods can be applied to inputs other than $\mathbb{R}^d$.

⚠️ $K(x, x') = \tanh(x^T x')$ is *not* always PSD, despite being used in practice!

---

[1]Matérn kernels are frequently used in the setting of spatial statistics and for Gaussian processes.

Illustration
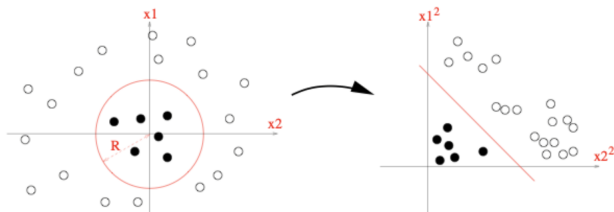
For $x = (x_1, x_2) \in \mathbb{R}^2$, let $\phi(x) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2) \in \mathbb{R}^3$, then

$$k(x, x') = \phi(x)^T \phi(x') = x_1^2 x_1'^2 + 2x_1 x_2 x_1' x_2' + x_2^2 x_2'^2 = (x^T x')^2.$$

The kernel $k(x, x')$ measures similarity between x and x' in $\mathbb{R}^2$, but computed implicitly via the more expressive features $\phi(x)$ in $\mathbb{R}^3$.

❓ It is easy to check that this particular kernel is PD using the definition. But how can we easily check whether the other kernels in the Examples are PD?

⚠ Remark on the Word "Kernel"

▶ **Smoothing/density estimation in Lecture 5:** A kernel is a weighting function for local averaging, e.g., for the Nadaraya-Watson estimator:

$$\hat{f}(x) = \frac{\sum_i K_h(x - x_i)y_i}{\sum_j K_h(x - x_j)}, \quad K_h(u) = \exp\left(-\frac{u^2}{2h^2}\right).$$

In general, the weighting function K_h is not required to be PD. It is typical to assume $K_h \geq 0$ and localized (Gaussian, Epanechnikov, etc.), so it controls how nearby data points are weighted.

▶ **Kernel methods (SVM, kernel ridge regression) in this lecture:** A valid kernel is a PD similarity function, e.g.

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right),$$

which corresponds to an inner product in a feature space.

Same word, different meanings — due to an unfortunate historical overlap in terminology.

Building New Kernels from Old

Now, let's go back to PD kernels. The following results are useful to build new kernels from old ones.

Proposition 1: Closure Properties of PD Kernels

Suppose $k_1, k_2, \ldots$ are PD kernels. Then the following are also PD kernels:

1. **Scaling:** $k(x, x') = \alpha k_1(x, x')$ for any $\alpha > 0$.
2. **Addition:** $k(x, x') = k_1(x, x') + k_2(x, x')$.
3. **Normalization:** $k(x, x') = g(x)k_1(x, x')g(x')$ for any real-valued function g.
4. **Limit:** $k(x, x') = \lim_{n \to \infty} k_n(x, x')$, if the limit exists for all x, x'.
5. **Product:** $k(x, x') = k_1(x, x')k_2(x, x')$.

Proof.
See blackboard. $\qquad \square$

Using the above results, we can show that the all the kernels in the Examples are PD (do it for the polynomial and the Gaussian kernels).

The Kernel Trick Illustrated

Let's derive the explicit feature map for the Gaussian (RBF) kernel in 1D to see why using kernels is so powerful. See Exercise 6.4 for the polynomial kernel.

$$
\begin{aligned}
k(x, x') &= \exp\left(-\frac{1}{2\sigma^2}(x - x')^2\right) \\
&= \exp\left(-\frac{x^2}{2\sigma^2}\right) \exp\left(\frac{xx'}{\sigma^2}\right) \exp\left(-\frac{x'^2}{2\sigma^2}\right) \\
&= \exp\left(-\frac{x^2}{2\sigma^2}\right) \left(\sum_{m=0}^{\infty} \frac{(xx'/\sigma^2)^m}{m!}\right) \exp\left(-\frac{x'^2}{2\sigma^2}\right) \\
&= \phi(x)^T \phi(x')
\end{aligned}
$$

The corresponding feature map $\phi(x)$ is:

$$
\phi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \left(1, \frac{x}{\sigma}, \frac{x^2}{\sigma^2\sqrt{2!}}, \frac{x^3}{\sigma^3\sqrt{3!}}, \dots\right)^T
$$

💡 The feature map $\phi(x)$ is infinite-dimensional. Computing it explicitly is impossible, but the kernel $k(x, x')$ can be computed easily. The kernel trick makes it possible to use this infinite-dim feature map!

Kernel Ridge Regression (KRR)

Given any PD kernel k (e.g., the Gaussian kernel), we have a corresponding hypothesis space:

$$\mathcal{H}_k = \left\{ f : f(x) = \sum_{j=1}^{\infty} \beta_j \phi_j(x), \text{ where } k(x, x') = \sum_{j=1}^{\infty} \phi_j(x)\phi_j(x') \right\}.$$

In this space, the regularized ERM has the solution we derived earlier, now expressed purely in terms of the kernel.

Definition 2: Kernel Ridge Regression

The predictor is given by:

$$\hat{f}(x) = \sum_{i=1}^{N} [(K + \lambda I_N)^{-1} y]_i k(x, x_i)$$

where K is the Gram matrix with $K_{ij} = k(x_i, x_j)$. This can be computed without any explicit knowledge of the feature maps $\{\phi_j\}$.

Learning Infinite-Dimensional Models in General

Dealing with infinite-dimensional models initially seems impossible because algorithms cannot be run in infinite dimensions. We have seen that the kernel function plays a crucial role in making such problems computationally tractable. Let's take a deeper look at this.

Learning with Linear Models in a Hilbert Space $\mathcal{H}$

Given data $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, for $i = 1, \ldots, N$, we consider the optimization problem:

$$\min_{\beta \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \langle \phi(x_i), \beta \rangle) + \frac{\lambda}{2} \|\beta\|^2,$$

where $\phi : \mathcal{X} \to \mathcal{H}$ is a (potentially infinite-dimensional) feature map that lift the input to the feature space $\mathcal{H}$, the loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$, the inner products and norms are taken w.r.t. the Hilbertian structure of $\mathcal{H}$.

The Representer Theorem

The key property of the objective function is that it only accesses inputs through dot products $\langle \beta, \phi(x_i) \rangle$ and penalizes the norm $\|\beta\|$.

Theorem 1: Representer Theorem

Let $\phi : \mathcal{X} \to \mathcal{H}$ be a feature map and $\Psi : \mathbb{R}^{N+1} \to \mathbb{R}$ be a functional that is strictly increasing in its last argument. Then any minimizer β^* of

$$\Psi(\langle \beta, \phi(x_1) \rangle, \ldots, \langle \beta, \phi(x_N) \rangle, \|\beta\|^2)$$

must lie in the span of the feature vectors of the training data. That is, β^* must be of the form:

$$\beta^* = \sum_{i=1}^{N} \alpha_i \phi(x_i), \quad \text{for some } \alpha \in \mathbb{R}^N.$$

Proof.
See blackboard.

$\square$

Representer Theorem for Supervised Learning

We can deduce the following result from the previous theorem.

Theorem 2: Representer Theorem for Supervised Learning

For any $\lambda > 0$, any minimizer β^* of the regularized risk

$$\frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \langle \beta, \phi(x_i) \rangle) + \frac{\lambda}{2} \|\beta\|^2$$

can be expressed as a linear combination of the feature vectors:

$$\beta^* = \sum_{i=1}^{N} \alpha_i \phi(x_i), \quad \text{for some } \alpha \in \mathbb{R}^N.$$

💡 There is no assumption on the loss function ℓ. In particular, no convexity is assumed. This result is very general.

The representer theorem can be extended to an interpolating estimator with essentially the same proof (see Exercise 6.2).

Reformulation using Kernels

Given the theorem, we can reformulate the learning problem by substituting $\beta = \sum_{i=1}^{N} \alpha_i \phi(x_i)$.

▶ Define the **kernel function** as the inner product of feature vectors:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle.$$

▶ The model's predictions on the training data become:

$$\langle \beta, \phi(x_j) \rangle = \sum_{i=1}^{N} \alpha_i \langle \phi(x_i), \phi(x_j) \rangle = \sum_{i=1}^{N} \alpha_i k(x_i, x_j) = (K\alpha)_j$$

where $K \in \mathbb{R}^{N \times N}$ is the kernel (Gram) matrix of the feature vector.

▶ The regularization term becomes a quadratic form in α:

$$\|\beta\|^2 = \left\langle \sum_i \alpha_i \phi(x_i), \sum_j \alpha_j \phi(x_j) \right\rangle = \sum_{i,j} \alpha_i \alpha_j K_{ij} = \alpha^\top K \alpha.$$

The Dual Problem and Prediction

The original infinite-dimensional problem in $\beta \in \mathcal{H}$ is now a finite-dimensional problem in $\alpha \in \mathbb{R}^N$:

$$\min_{\alpha \in \mathbb{R}^N} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, (K\alpha)_i) + \frac{\lambda}{2} \alpha^\top K \alpha.$$

For any test point $x \in \mathcal{X}$, the prediction function also only requires kernel evaluations:

$$f(x) = \langle \beta, \phi(x) \rangle = \sum_{i=1}^{N} \alpha_i \langle \phi(x_i), \phi(x) \rangle = \sum_{i=1}^{N} \alpha_i k(x, x_i).$$

🔔 Convince youself that the smoothing splines (Lecture 5) are a special case of kernel methods, with a specific kernel derived from certain smoothness penalty.

The Kernel Trick: Summary

💡 The input observations are summarized in the $N \times N$ kernel matrix K. We never need to explicitly compute the feature vector $\phi(x)$.

This is the **kernel trick**, which allows us to:

- ▶ **Solve in $\mathbb{R}^N$:** Replace the potentially infinite-dimensional search space $\mathcal{H}$ with the finite-dimensional space $\mathbb{R}^N$.
- ▶ **Separate Concerns:** Separate the representation problem (designing powerful kernels $k(\cdot, \cdot)$ for many data types) from the algorithmic problem (which only uses the Gram matrix K).

Computational Implications

The infinite-dimensional optimization problem reduces to the finite-dimensional problem:

$$\min_{\alpha \in \mathbb{R}^n} \left[\sum_{i=1}^{N} \ell \left(y_i, \sum_{j=1}^{N} \alpha_j K(x_i, x_j) \right) + \lambda \sum_{i,j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) \right],$$

which can be written as: $\min_{\alpha \in \mathbb{R}^N} \left[\sum_{i=1}^{N} \ell(y_i, (K\alpha)_i) + \lambda \alpha^T K \alpha \right]$.

▶ **Finite computation:** Only need to work with $N \times N$ Gram matrix K

▶ **No explicit features:** Never compute $\phi(x)$, only kernel evaluations

▶ **General applicability:** Works for any loss function ℓ

▶ **Scales with data:** Complexity is $O(N)$, not dimension of feature space

The Computational Bottleneck of Kernel Methods

⚠ The Representer Theorem is powerful, but it leads to a significant computational challenge for large datasets.

The Problem: The Gram Matrix
The solution for kernel methods like Kernel Ridge Regression or SVMs involves solving a linear system with the $N \times N$ Gram matrix K.

$$\boldsymbol{\alpha} = (K + \lambda \mathbf{I})^{-1} \mathbf{y} \quad \text{(for Kernel Ridge Regression)}.$$

▶ Storage Cost: Storing the Gram matrix requires $O(N^2)$ memory.

▶ Computational Cost: Solving the linear system (e.g., via matrix inversion or Cholesky decomposition) costs $O(N^3)$ time. This is infeasible for large N.

Kernel methods in their standard form **do not scale well** with the number of samples N. We need approximation techniques.

The Nyström Approximation Method

Idea: Approximate K by sampling $D \ll N$ *anchor points* from the dataset.

▶ Select $D \ll N$ anchor points randomly to obtain: $\mathcal{A} = \{x_{i_1}, \ldots, x_{i_D}\}$.

▶ Define kernel submatrices:

$$K_{D,D} = [k(x_i, x_j)]_{x_i, x_j \in \mathcal{A}}, \quad K_{N,D} = [k(x_i, x_j)]_{i=1,\ldots,N;\, x_j \in \mathcal{A}}.$$

▶ **Nyström approximation:** $K \approx K_{N,D} K_{D,D}^{-1} K_{N,D}^{\top}$.

▶ Predictor[2] for a new point x: $f(x) \approx \sum_{j=1}^{D} \alpha_j k(x, x_{i_j})$.

▶ Cost: $O(ND^2)$, much less than $O(N^3)$ for exact KRR. This is a huge savings when $D \ll N$.

💡 Low-rank matrix approximations are essential tools in the application of kernel methods to large-scale learning problems, and Nyström's method is one of those.

⚠ These methods are powerful, but they work purely at the *algorithmic* level. What if we could exploit the *mathematical structure* of the kernel itself?

---

[2]The coefficients α_j are found by solving the reduced $D \times D$ system with the Nyström approximation.

Approximation via Random Fourier Features

For shift-invariant kernels, Fourier analysis provides a elegant alternative.

Theorem 3: Bochner's Theorem (Special Case)

Let $q : \mathbb{R}^d \to \mathbb{R}$ be continuous. A shift-invariant kernel $k(x, x') = q(x - x')$ is PD if and only if it is the Fourier transform of a non-negative finite measure μ. If μ admits a density $p(\omega)$ with respect to Lebesgue measure, then

$$q(x - x') = \int_{\mathbb{R}^d} p(\omega) \, e^{i\omega^\top (x - x')} \, d\omega, \quad p(\omega) \geq 0.$$

If moreover $\int_{\mathbb{R}^d} p(\omega) \, d\omega = 1$, then p is a probability density.

Proof.
See, e.g., Bach Ch. 7.3.3. □

💡 Bochner's theorem allows us to interpret the similarity measure in the frequency domain. We can approximate the integral with a Monte Carlo average. By sampling D frequencies ω_j from the density $p(\omega)$, we can construct an explicit, low-dim feature map that approximates the infinite-dim map.

The Random Fourier Features (RFFs)

Random Fourier Features for Gaussian Kernels

1: **Goal:** Approximate the Gaussian kernel $k(x, x') = \exp(-\frac{\|x-x'\|^2}{2\sigma^2})$. Its Fourier transform is a Gaussian density.
2: **Sampling:** Draw D random vectors $\omega_1, \ldots, \omega_D$ from $\mathcal{N}(0, \frac{1}{\sigma^2}\mathbf{I})$.
3: **Feature Map:** Create the new feature map $z : \mathbb{R}^d \to \mathbb{R}^{2D}$:

$$z(x) = \frac{1}{\sqrt{D}} \begin{bmatrix} \cos(\omega_1^T x) \\ \sin(\omega_1^T x) \\ \vdots \\ \cos(\omega_D^T x) \\ \sin(\omega_D^T x) \end{bmatrix} \in \mathbb{R}^{2D}.$$

4: **Linear Model:** Train a standard linear model (e.g., Ridge Regression or a linear SVC) on the new data $(z(x_i), y_i)$.

The inner product of these new features, $z(x)^T z(x')$, is an unbiased estimator of the true kernel value $k(x, x')$ (see Exercise 6.5).

Theoretical Guarantees for Random Fourier Features

The approximation error of the estimator decreasing as D increases.

Theorem 4: Uniform Approximation for Gaussian Kernels

For any dataset $\{x_1, \ldots, x_N\}$ and $\varepsilon, \delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\max_{i,j \leq N} \left| z(x_i)^\top z(x_j) - k(x_i, x_j) \right| \leq \varepsilon$$

provided $D \geq \frac{2}{\varepsilon^2} \log\left(\frac{2N^2}{\delta}\right)$.

Proof.
See blackboard. □

💡 Training cost[3] with random features is $O(ND^2 + D^3)$ compared to $O(N^3)$ for exact kernel methods. This is a huge savings when $D \ll N$.

---

[3]The D^3 term comes from solving the regularized linear system in the transformed feature space.

Nyström Method vs. Random Fourier Features

Aspect	Nyström	RFFs
Anchors	Randomly sampled data points	Random frequencies (Monte Carlo)
Approx.	Low-rank kernel matrix	Fourier expansion of kernel
Kernel type	Any kernel	Shift-invariant kernels
Cost	$O(ND^2)$	$O(ND^2 + D^3)$
Predictor	$f(x) \approx \sum_{j=1}^{D} \alpha_j k(x, x_{i_j})$	$f(x) \approx \beta^\top z(x)$, $z =$ feature map

💡 Both reduce $O(N^3)$ training to something feasible when $D \ll N$.

💡 **Rule of thumb:** Use RFF for shift-invariant kernels when you need explicit features and numerical stability, and Nyström for general kernels or when computational efficiency is paramount.

Example: Approximating a Kernel Machine with RFFs

We can run a simulation to see how well the random features approximation works in practice.

- ▶ **Data:** A non-linearly separable "two moons" dataset with $N = 500$ points.
- ▶ **Models:**
    1. An exact Kernel Ridge Regression classifier (our gold standard).
    2. Several linear Ridge Regression classifiers trained on Random Fourier Features, with an increasing number of features D.
- ▶ **Analysis:** We will plot the test accuracy of the approximate models as a function of the number of random features D and compare it to the accuracy of the exact kernel machine.

Example: Random Features in Action
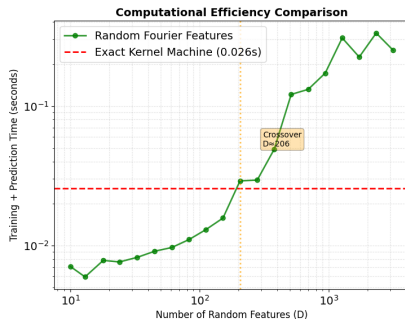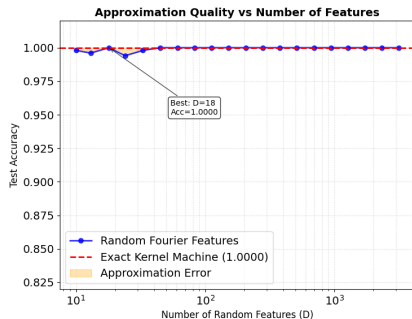
Random features can achieve nearly the same predictive power as a full kernel machine at a fraction of the computational cost, making kernel methods practical for large datasets!

💡 The seminal work by Rahimi and Recht on random features[4] won the 2017 NIPS Test of Time Award:
`https://eecs.berkeley.edu/news/ben-recht-wins-nips-test-time-award/`

---

[4]Random features are cool stuffs. See, e.g., `https://arxiv.org/abs/2004.11154` for a survey.

Exercises

Exercise 6

1. (a) Show that if $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive-definite kernel, then so is the function $(x, x') \mapsto e^{k(x,x')}$.
   (b) Write down the explicit expressions for the Matérn kernel when $\nu = 3/2$ and $\nu = 5/2$. Verify that the resulting kernels for these two cases are positive definite.

2. **Minimum Norm Interpolation.** Let $\mathcal{H}$ be a Hilbert space with feature map $\phi : \mathcal{X} \to \mathcal{H}$. Given data $(x_1, y_1), \ldots, (x_N, y_N)$ where $x_i \in \mathcal{X}$ and $\mathbf{y} \in \mathbb{R}^N$. Assume that the set of interpolating solutions is non-empty, i.e., there exists at least one $\beta \in \mathcal{H}$ such that $y_i = \langle \beta, \phi(x_i) \rangle_{\mathcal{H}}$ for all $i \in \{1, \ldots, N\}$. Show that, among all such interpolating solutions, the unique solution β^* with the minimum norm $\|\beta^*\|_{\mathcal{H}}$ can be expressed as: $\beta^* = \sum_{i=1}^{N} \alpha_i \phi(x_i)$, where the coefficient vector $\alpha \in \mathbb{R}^N$ is a solution to the linear system $\mathbf{y} = K\alpha$. Here, K is the $N \times N$ Gram matrix with entries $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$.

Exercises

Exercise 6

3. The primal problem for Kernel Ridge Regression (KRR) is to find the vector $\beta \in \mathbb{R}^M$ in a feature space of dimension M that minimizes:

$$L_P(\beta) = \|\mathbf{y} - \Phi\beta\|_2^2 + \lambda\|\beta\|_2^2,$$

where $\Phi \in \mathbb{R}^{N \times M}$ is the design matrix of feature vectors. This leads to solving a linear system involving the $M \times M$ primal matrix $H_P = \Phi^\top \Phi + \lambda I_M$.

(a) Derive the dual problem for KRR. Show that it involves solving a linear system for a dual variable $\alpha \in \mathbb{R}^N$ with the $N \times N$ dual matrix $H_D = K + \lambda I_N$, where $K = \Phi\Phi^\top$ is the kernel Gram matrix.

(b) The condition number of a matrix measures the numerical stability of a linear system, with lower values being better. Compare the condition number of the primal matrix H_P with that of the dual matrix H_D. Analyze the two cases: (i) $N > M$, (ii) $M > N$ (the typical scenario in KRR).

(c) Compare the two formulations to the use of normal equations as in Lecture 2, and relate the two using the matrix inversion lemma.

Exercises

Exercise 6

4. Show that the polynomial kernel $k(x, x') = (1 + x^\top x')^p$ for $x, x' \in \mathbb{R}^d$ corresponds to a feature space spanned by all monomials of the form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$ such that the total degree $\sum_{i=1}^d \alpha_i \leq p$. Also, show that the dimension of this feature space is given by the binomial coefficient: $\binom{d+p}{p}$.

5. (a) **Gaussian Kernel (RFF).** Let $\omega_j \overset{\text{iid}}{\sim} \mathcal{N}(0, \frac{1}{\sigma^2} I)$ and define $z(x) = \frac{1}{\sqrt{D}} \big(\cos(\omega_1^\top x), \sin(\omega_1^\top x), \ldots, \cos(\omega_D^\top x), \sin(\omega_D^\top x) \big)$, with estimator $\hat{k}_D^{\text{RFF}}(x, x') := z(x)^\top z(x')$.

   i. Show that $\mathbb{E}[\hat{k}_D^{\text{RFF}}(x, x')] = k_{\text{RBF}}(x, x') = \exp(-\|x - x'\|^2 / 2\sigma^2)$.

   ii. Derive $\text{Var}[\hat{k}_D^{\text{RFF}}(x, x')]$ in terms of $k(x, x')$ and D. What is the scaling with D?

   (b) **Softmax Kernel[a] (Positive Random Features).** For $w_j \overset{\text{iid}}{\sim} \mathcal{N}(0, I)$, define $\varphi_w(z) = \exp(-\|z\|^2/2 + w^\top z)$ and estimator $\hat{k}_D^{\text{soft}}(x, x') := \frac{1}{D} \sum_{j=1}^D \varphi_{w_j}(x) \varphi_{w_j}(x')$.

   i. Show that $\mathbb{E}[\hat{k}_D^{\text{soft}}(x, x')] = k_{\text{soft}}(x, x') = \exp(x^\top x')$.

   ii. Derive $\text{Var}[\hat{k}_D^{\text{soft}}(x, x')]$ in terms of x, x' and D. Give the scaling with D.

---

[a]Cool stuff! This kernel was considered for efficient estimation of (softmax) full-rank-attention Transformers with provable accuracy; see the Performers paper by Choromanski et. al. (ICLR 2021).

Exercises

Exercise 6 (Additional: Bayesian Interpretation of KRR)

Consider a linear model in a feature space, $f(\mathbf{x}) = \phi(\mathbf{x})^\top \beta$, with a Bayesian setup:

- **Likelihood:** $p(\mathbf{y}|\Phi, \beta, \sigma^2) = \mathcal{N}(\mathbf{y}|\Phi\beta, \sigma^2 I_N)$.
- **Gaussian Prior:** $p(\beta|\sigma_\beta^2) = \mathcal{N}(\beta|\mathbf{0}, \sigma_\beta^2 I_M)$.

Goal: We want to show that finding the Maximum A Posteriori (MAP) estimate for β is equivalent to solving kernel ridge regression (KRR).

(a) Show that minimizing the negative log-posterior, $-\ln p(\beta|\mathbf{y}, \Phi)$, yields the MAP objective: $J_{\text{MAP}}(\beta) \propto \frac{1}{2\sigma^2}\|\mathbf{y} - \Phi\beta\|_2^2 + \frac{1}{2\sigma_\beta^2}\|\beta\|_2^2$.

(b) State the standard KRR objective, $J_{\text{KRR}}(\beta)$, which combines the sum-of-squared-errors loss with an L2 regularization penalty λ. By comparing J_{MAP} and J_{KRR}, show their equivalence and derive the expression for the regularization parameter λ in terms of the model variances σ^2 and σ_β^2.

(c) Interpret your result for λ. What does a large λ imply about the data noise variance (σ^2) relative to the prior weight variance (σ_β^2)?

(d) Show that the MAP solution can be expressed as $\beta^* = \sum_{i=1}^{N} \alpha_i \phi(x_i)$, where $\alpha = (K + \lambda_{MAP} I_N)^{-1}\mathbf{y}$ for some λ_{MAP} and Gram matrix K.

Exercises

Exercise 6 (Experimental)

How does a kernel's performance depend on whether its implicit smoothness assumption matches the regularity of the target function?

▶ **Setup:** You will perform Kernel Ridge Regression (KRR) for two functions defined on $[0, 1]$.

  ▶ **Smooth Function (Sine Wave):** $f_{\text{smooth}}(x) = \sin(4\pi x)$
  ▶ **Non-smooth Function (Piecewise Constant, Square Wave):**
  $$f_{\text{nonsmooth}}(x) = \begin{cases} 1 & \text{if } 0 \leq x < 0.25, 0.5 \leq x < 0.75 \\ -1 & \text{otherwise} \end{cases}$$

▶ **Kernels to Compare:** You will compare three kernels with different smoothness properties.

  ▶ **Low Smoothness (Exponential):** $k(x, x') = \exp\left(-\frac{|x-x'|}{\ell}\right)$
  ▶ **Medium Smoothness (Matérn 5/2):**
  $k(x, x') = \left(1 + \frac{\sqrt{5}|x-x'|}{\ell} + \frac{5(x-x')^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}|x-x'|}{\ell}\right)$
  ▶ **High Smoothness (Gaussian/RBF):** $k(x, x') = \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$

Exercises

Exercise 6 (Experimental)

▶ **Procedure:** For *each* of the two target functions:

  ▶ Generate a training set of $N = 40$ points by sampling $x_i \sim$ Uniform$[0, 1]$ and setting $y_i = f(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 0.1^2)$.
  ▶ For *each* of the three kernels, train a KRR model.
  ▶ **Crucially**, tune the hyperparameters (length-scale ℓ and regularization λ) for each model using 5-fold cross-validation on the training set to ensure a fair comparison.

▶ **Evaluation and Analysis:**

  ▶ For each of the six final models (2 functions $\times$ 3 tuned kernels), compute the mean squared error (MSE) on a set of 1000 uniformly spaced points spanning $[0, 1]$. Create a summary table for the six test MSE values.
  ▶ Create two plots (one for the sine wave, one for the square wave) showing the true function, the noisy data, and the three fitted KRR curves.
  ▶ Discuss the results. Which kernel performed best on the smooth sine function and which on the non-smooth square wave?

Appendix: From Kernels to Function Spaces

Here, we collect basic theoretical results of kernel methods and RKHS[5]. For full details, refer to, e.g., Ch. 7 of Bach or Hofmann-Schölkopf-Smola (Kernel methods in machine learning, 2008): https://arxiv.org/abs/math/0701907.

1. **Finite-Dim Feature Construction from a Kernel**: with linear algebra
2. **Mercer's Theorem**: spectral view of kernels (explicit construction of feature maps via eigen-expansion).
3. **Aronszajn's Theorem**: abstract construction of kernels as inner products in some Hilbert space, justifies the kernel trick.
4. **Reproducing Kernel Hilbert Space (RKHS)**: definition, key properties.
5. **Moore–Aronszajn Theorem**: existence and uniqueness of RKHS.
6. **Examples**: linear, polynomial, Gaussian kernels.
7. **Representer Theorem**: why kernel methods are computationally feasible.

---

[5]This is an optional topic and will not be tested in the exam.

Finite-Dimensional Feature Construction from a Kernel

Goal: Given a PD kernel k and finite points $x_1, \ldots, x_N \in \mathcal{X}$, construct explicit feature vectors $\phi(x_i)$ such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$.

1. Form the Gram matrix: $K = [k(x_i, x_j)]_{i,j=1}^N \in \mathbb{R}^{N \times N}$
2. Compute eigendecomposition (always valid for PSD matrices):

$$K = U\Lambda U^\top, \quad \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_N), \ \lambda_i \geq 0$$

3. Define feature vectors:

$$\phi(x_i) = \Lambda^{1/2} U^\top e_i \in \mathbb{R}^N$$

where e_i is the i-th standard basis vector.

4. Verify:

$$\langle \phi(x_i), \phi(x_j) \rangle = \phi(x_i)^\top \phi(x_j) = K_{ij} = k(x_i, x_j)$$

💡 For any finite dataset, the kernel can always be written as an inner product in $\mathbb{R}^N$. The eigen-decomposition of the Gram matrix K provides explicit feature vectors. This construction motivates the spectral decomposition of kernels in the general (possibly infinite) domain case.

Mercer's Theorem

The finite-dimensional construction suggests a natural spectral decomposition for kernels. If the kernel k is continuous, PD on a compact domain, then we can explicitly construct (possibly infinite-dim) feature map via eigenfunctions.

Theorem 5: Mercer's Theorem (Simplified)

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a continuous, positive definite kernel on a compact domain $\mathcal{X}$. Then there exist eigenvalues $\{\lambda_i \geq 0\}$ and orthonormal eigenfunctions $\{\psi_i\}$ of the integral operator

$$(T_k f)(x) = \int_{\mathcal{X}} k(x, x') f(x') dx',$$

such that $k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x')$ (uniformly and absolutely convergent).

💡 This gives an implicit feature map: $\phi(x) = (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \dots)$ so that $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\ell^2}$.

Aronszajn's Theorem

What if we do not assume compactness or continuity? In this case, we do not have an explicit construction of the feature map, but rather an abstract construction that guarantees the existence of a Hilbert space where we can construct the kernels as inner products.

Theorem 6: Aronszajn's Theorem

A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite (PD) kernel if and only if there exists a Hilbert space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}.$$

Proof.
See, e.g., Proposition 7.3 in Bach. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

💡 This theorem justifies the kernel trick!

Reproducing Kernel Hilbert Space (RKHS)

We can understand kernels as defining function spaces with certain properties.

Definition 3: Reproducing Kernel Hilbert Space

Given a PD kernel k on domain $\mathcal{X}$, the **Reproducing Kernel Hilbert Space** $\mathcal{H}_k$ is a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ with two key properties:

1. **Kernel functions are in the space:** For each $x \in \mathcal{X}$, the function $k_x(\cdot) := k(\cdot, x)$ belongs to $\mathcal{H}_k$.

2. **Reproducing property:** For any $f \in \mathcal{H}_k$ and $x \in \mathcal{X}$:

$$f(x) = \langle f, k_x \rangle_{\mathcal{H}_k} = \langle f, k(\cdot, x) \rangle_{\mathcal{H}_k}.$$

💡 The RKHS is the "natural" function space associated with kernel k. Functions in $\mathcal{H}_k$ can be evaluated at any point via inner product with the kernel.

The Moore-Aronszajn Theorem

Aronszajn's theorem only guarantees that some Hilbert space exists where k can be realized as an inner product. In fact, there exists a minimal, unique Hilbert space — the RKHS associated with k.

Theorem 7: Existence and Uniqueness of RKHS

For every PD kernel k on $\mathcal{X}$, there exists a unique Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ having k as its reproducing kernel.

In RKHS, evaluating a function at a point is the same as taking an inner product with the kernel function at that point.

Proposition 2: Key Properties

- $k(x, y) = \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}_k}$.
- For $f = \sum_i \alpha_i k(\cdot, x_i)$, $\|f\|_{\mathcal{H}_k}^2 = \sum_i \sum_j \alpha_i \alpha_j k(x_i, x_j)$.

In short: kernels encode inner products without explicitly computing feature maps.

RKHS Examples

Linear Kernel: $k(x, x') = x^T x'$

$\mathcal{H}_k = \{f(x) = w^T x : w \in \mathbb{R}^d\}$ with $\|f\|^2_{\mathcal{H}_k} = \|w\|^2$
This recovers standard linear regression/classification.

Polynomial Kernel: $k(x, x') = (1 + x^T x')^p$

$\mathcal{H}_k$ consists of polynomials up to degree p with appropriate norm.

Gaussian Kernel: $k(x, x') = \exp(-\|x - x'\|^2/(2\sigma^2))$

$\mathcal{H}_k$ is infinite-dimensional and very rich:

► Contains smooth functions
► Functions decay appropriately at infinity

> 💡 Different kernels encode different notions of function complexity via their RKHS norms.

The Key Result for Kernel Methods

Theorem 8: The Representer Theorem

Consider the regularized empirical risk minimization problem:

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

where L is any loss function, $\lambda > 0$, and $\mathcal{H}_k$ is the RKHS associated with PD kernel k. Then the minimizer $\hat{f}$ has the finite representation: $\hat{f}(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i)$ for some coefficients $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$.

💡 **Remarkable:** Even though the optimization is over an infinite-dimensional RKHS, the solution lives in the finite-dimensional subspace $\text{span}\{k(\cdot, x_1), \ldots, k(\cdot, x_N)\}$. Thus, learning reduces to finding N coefficients $\alpha_1, \ldots, \alpha_N$.

Summary: The Power of the Kernel Viewpoint

Kernel methods are the natural generalization of (l_2-regularized) linear models to more general spaces.

- ▶ **Generality:** The kernel trick provides a powerful recipe for converting linear techniques that depend on inner products into non-linear methods capable of learning complex decision boundaries or regression functions.

- ▶ **Computational Efficiency:** By working with the Gram matrix K, the complexity of the algorithm depends on the number of samples, not the (potentially infinite) dimension of the feature space.

- ▶ **Theoretical Foundation:** The theory of RKHS provides a rigorous mathematical framework for understanding why these methods work and for designing new kernel functions.

- ▶ **Unifying Framework:** The "Loss + Penalty in an RKHS" formulation, justified by the Representer Theorem, unifies many disparate-seeming methods (SVMs, Ridge Regression, Smoothing Splines) under a single elegant theory.