Mathematical Introduction to Machine Learning

Lecture 2: Linear Methods for Regression

September 26, 2025

Lecturer: Lei Wu *Scribe: Lei Wu*

In this chapter, we introduce several popular linear methods for regression problem, which have been widely used in practice. The advantage of linear methods is that they always have good theoretical guarantees due to the simplicity.

1 Linear regression

Linear regression is the simplest method in statistics and machine learning, and it often serves as a good illustrative example for understanding machine learning models and algorithms.

The hypothesis space of linear regression is given by

$$\mathcal{H} = \left\{ \mathbf{x} \mapsto \boldsymbol{\beta}^\top \mathbf{x} + \beta_0 \,:\, \boldsymbol{\beta} \in \mathbb{R}^d, \beta_0 \in \mathbb{R} \right\}.$$

In this case, $\theta = (\boldsymbol{\beta}, \beta_0)$ are the parameters to be learned from data. It is also customary to introduce the extended coordinate $\tilde{\mathbf{x}} = (\mathbf{x}^\top, 1)^\top \in \mathbb{R}^{d+1}$ and let $\tilde{\boldsymbol{\beta}} = (\boldsymbol{\beta}^\top, \beta_0)^\top \in \mathbb{R}^{d+1}$. Then, we can write $\boldsymbol{\beta}^\top \mathbf{x} + \beta_0 = \tilde{\boldsymbol{\beta}}^\top \tilde{\mathbf{x}}$. Hence, it is often simply to write

$$\mathcal{H} = \left\{ \mathbf{x} \mapsto \boldsymbol{\beta}^\top \mathbf{x} \,:\, \boldsymbol{\beta} \in \mathbb{R}^d \right\}.$$

Given the data set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the empirical risk is given by

$$\widehat{\mathcal{R}}(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \left(\boldsymbol{\beta}^\top \mathbf{x}_i - y_i \right)^2 = \frac{1}{2n} \left\| X\boldsymbol{\beta} - \mathbf{y} \right\|_2^2. \tag{1}$$

Here, $X = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times d}$ be the data matrix and $\mathbf{y} = (y_1, y_2, \cdots, y_n)^\top \in \mathbb{R}^n$.

1.1 Ordinary Least Squares (OLS)

In ordinary least squares (OLS), the estimator is defined as the minimizer of the empirical risk (1). Since $\widehat{\mathcal{R}}$ is a quadratic function of $\boldsymbol{\beta}$, its minimizer can be obtained by setting the gradient to zero:

$$\nabla \widehat{\mathcal{R}}(\boldsymbol{\beta}) \;=\; \frac{1}{n} X^\top \left(X\boldsymbol{\beta} - \mathbf{y} \right) \;=\; 0.$$

This yields

$$\left(X^\top X \right) \boldsymbol{\beta} \;=\; X^\top \mathbf{y}, \tag{2}$$

which is known as the *normal equation*. Here $X^\top X$ is called the *Gram matrix*.

If $X^\top X$ is full rank, the OLS estimator admits the closed-form solution

$$\hat{\boldsymbol{\beta}}_{\text{ols}} \;=\; \left(X^\top X \right)^{-1} X^\top \mathbf{y}. \tag{3}$$

When $X^\top X$ is singular—for instance, in the over-parameterized regime $d > n$—the minimizer is not unique. In such cases, which solution should we choose? From a statistical perspective, we prefer a solution with small complexity that generalizes well. A common choice is the *minimum-norm solution*:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{\beta}\|_c \\ \text{s.t.} \quad & X\boldsymbol{\beta} = \mathbf{y}, \end{aligned} \tag{4}$$

where the choice of norm $\|\cdot\|_c$ depends on the context. In practice, the ℓ_2 norm is most frequently used. In this case, if $\mathrm{rank}(X) = n$, the minimum-ℓ_2-norm solution admits the closed-form expression

$$\hat{\boldsymbol{\beta}}_{\ell_2} \;=\; X^\top (XX^\top)^{-1}\mathbf{y}. \tag{5}$$

Exercise 1.1. Let X^+ denote the Moore–Penrose pseudoinverse of X. Prove that the minimum-ℓ_2-norm solution can be written as

$$\hat{\boldsymbol{\beta}}_{\ell_2} \;=\; X^+\mathbf{y}.$$

Moreover, show that if $\mathrm{rank}(X) = n$ (i.e., X has full row rank), then

$$X^+ \;=\; X^\top (XX^\top)^{-1}.$$

The above estimator forces the model to fit the training data exactly. This can be problematic when the data are noisy:

$$y_i \;=\; \boldsymbol{\beta}_*^\top \mathbf{x}_i + \xi_i, \qquad \xi_i \neq 0.$$

In such cases, the OLS estimator tends to overfit the noise, which in turn degrades generalization performance. To address this issue, a common strategy is to introduce *regularization*, leading to the following penalized empirical risk:

$$\frac{1}{2n}\|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 \;+\; \lambda\, r(\boldsymbol{\beta}).$$

Here $r(\boldsymbol{\beta})$ is a penalty term that encodes prior knowledge about the parameter $\boldsymbol{\beta}$, while $\lambda > 0$ is a hyperparameter controlling the trade-off between data fidelity and regularization.

This naturally raises two fundamental questions:

- How should we choose the form of the penalty $r(\cdot)$?
- How should we set the regularization parameter λ?

1.2 Ridge Regression

As a first example of regularization, we consider *ridge regularization*, which corresponds to choosing $r(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|_2^2$. Then, the objective function then becomes

$$\frac{1}{2n}\|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 \;+\; \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2. \tag{6}$$

A key advantage of this formulation is that the minimizer admits a closed-form solution:

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} \;=\; \left(\tfrac{1}{n}X^\top X + \lambda I_d\right)^{-1} \tfrac{1}{n} X^\top \mathbf{y}. \tag{7}$$

It is worth noting that $\frac{1}{n}X^\top X + \lambda I_d$ is always non-singular for any $\lambda > 0$, even if $X^\top X$ itself is singular. Thus, ridge regression guarantees a unique and stable solution, in contrast to ordinary least squares, which may fail in high-dimensional or ill-conditioned settings.

Question 1.2. Why do we regularize with the squared ℓ_2 norm $\frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2$ rather than the ℓ_2 norm $\frac{\lambda}{2}\|\boldsymbol{\beta}\|_2$?

More generally, ridge regularization can be viewed as a special case of *Tikhonov regularization*, where the objective takes the form

$$\frac{1}{2n}\|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 \;+\; \frac{\lambda}{2}\|\Gamma\boldsymbol{\beta}\|_2^2, \tag{8}$$

with $\Gamma \in \mathbb{R}^{d\times d}$ a prescribed weighted matrix that controls how different coordinates are penalized. Ridge regression corresponds to the simple case $\Gamma = I_d$.

Remark 1.3. Ridge regression is not only a fundamental statistical tool but also a building block for many machine learning methods: (1) It is the foundation of *kernel ridge regression*, which generalizes to nonlinear models through kernels. (2) It underlies *weight decay* in neural networks, a widely used regularization strategy. (3) It provides insight into how regularization stabilizes training, improves conditioning, and combats overfitting. Thus, ridge regression serves as a prototype illustrating how regularization shapes both theory and practice in modern machine learning.

Exercise 1.4. Let $\hat{\boldsymbol{\beta}}_\lambda$ denote the ridge regression solution in (6) for $\lambda > 0$. Show that as $\lambda \to 0$, the ridge estimator converges to the minimum-ℓ_2-norm solution of the least-squares problem, i.e.,

$$\hat{\boldsymbol{\beta}}_\lambda \;\to\; \hat{\boldsymbol{\beta}}_{\ell_2}.$$

1.3 Least absolute shrinkage and selection operator (Lasso)

Another widely used regularized linear model is the *Lasso*, defined as the solution to

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \frac{1}{2n}\|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 \;+\; \lambda\|\boldsymbol{\beta}\|_1. \tag{9}$$

Unlike ridge regression, which penalizes the ℓ_2 norm, Lasso imposes an ℓ_1 penalty. The key motivation for using the ℓ_1 norm is its ability to promote *sparsity* in the solution. To formalize this, we introduce the following assumption:

Assumption 1.5 (Sparsity)**.** *Let $\|\boldsymbol{\beta}\|_0 = \#\{i \in [d] \,:\, \beta_i \neq 0\}$, as the number of nonzero entries in $\boldsymbol{\beta}$. Assume that the ground-truth parameter $\boldsymbol{\beta}^*$ is sparse, i.e, $\|\boldsymbol{\beta}^*\|_0 \ll d$.*

This sparsity assumption is common in high-dimensional applications, where only a small subset of variables has a substantial effect. In such cases, our goal is not only to achieve accurate prediction but also to identify the few *relevant variables*. This leads to two main rationales for enforcing sparsity:

- **Generalization / Prediction.** In the high-dimensional regime where $d \gg n$, the number of variables far exceeds the number of samples, making generalization in general unattainable. However, if the true parameter $\boldsymbol{\beta}^*$ is sparse, with $\|\boldsymbol{\beta}^*\|_0 \ll n$, then reliable generalization becomes possible by exploiting this sparsity. Intuitively, sparsity reduces the effective complexity of the model, allowing one to learn from limited data while still capturing the essential structure of the target.

- **Interpretability.** In many applications, interpretability is as important as – if not more important than – prediction accuracy. For example, in medical studies of a certain disease, there may be many potential risk factors. A sparse model helps identify the most relevant ones, thereby guiding further scientific investigation. In such settings, the primary goal is *variable selection*, rather than maximizing predictive power.

Remark 1.6. Consider $\boldsymbol{\beta}^* = (1.4,\ 0.2,\ 0,\ 0,\ 0) \in \mathbb{R}^5$ and its two approximations:

$$\hat{\boldsymbol{\beta}}_1 = (1,\ 0.3,\ 0,\ 0,\ 0), \qquad \hat{\boldsymbol{\beta}}_2 = (1.4,\ 0.2,\ 0.15,\ 0.02,\ 0.05).$$

Although $\|\hat{\boldsymbol{\beta}}_2 - \boldsymbol{\beta}^*\|_2 < \|\hat{\boldsymbol{\beta}}_1 - \boldsymbol{\beta}^*\|_2$, one may still prefer $\hat{\boldsymbol{\beta}}_1$. The vector $\hat{\boldsymbol{\beta}}_1$ is sparse and exactly recovers the true support $\{1, 2\}$, clearly indicating that only the first two variables are relevant. In contrast, $\hat{\boldsymbol{\beta}}_2$ assigns small but nonzero coefficients to irrelevant coordinates, introducing spurious features and obscuring interpretability. This example highlights why sparsity-promoting penalties are valuable: they yield simpler, more interpretable models, even at the cost of a larger prediction error.

A natural formulation that directly promotes sparsity is

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \frac{1}{2n} \|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_0. \tag{10}$$

However, this optimization problem is computationally intractable because the ℓ_0 "norm" [1] is neither continuous nor convex. To overcome this difficulty, one can relax the ℓ_0 penalty to an ℓ_p penalty with $p > 0$:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \frac{1}{2n} \|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_p. \tag{11}$$

Figure 1 illustrates the ℓ_p landscapes in one dimension for various values of p. The ℓ_p penalty is continuous for all $p > 0$, but it is convex only when $p \geq 1$. Since convex optimization problems are typically much easier to solve than non-convex ones, it is natural that the case $p = 1$ is especially favored: it is the smallest p that ensures convexity along this relaxation path.

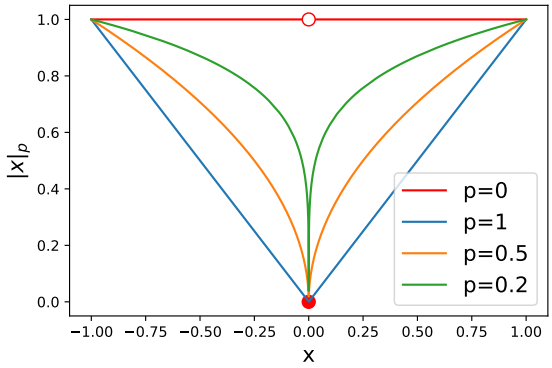

Figure 1: An illustration of the landscape of ℓ_p norm for various p's. Here, x is an one-dimensional variable.

However, the preceding intuition merely indicates that the ℓ_1 penalty serves as a surrogate for ℓ_0, and it is often expected that the corresponding solutions may be close. This, however, does not guarantee that the ℓ_1 solution will be exactly sparse, even when the ground-truth parameter $\boldsymbol{\beta}^*$ itself is sparse. See Remark 1.6 for an illustration of why closeness in ℓ_2 norm does not necessarily imply sparsity.

To gain geometric insight into how ℓ_p regularization can induce sparsity with $p \in (0, 1)$, it is useful to consider the equivalent constrained problem

$$\min_{\|\boldsymbol{\beta}\|_p \leq t} \frac{1}{2} \|X\boldsymbol{\beta} - \mathbf{y}\|_2^2. \tag{12}$$

[1] Here the term "norm" is used in a loose sense, merely as a measure of model complexity.

Figure 2 shows the contour lines of the empirical risk $\widehat{\mathcal{R}}(\boldsymbol{\beta}) = \frac{1}{2}\|X\boldsymbol{\beta} - \mathbf{y}\|_2^2$ (dashed curves) together with those of the regularizer $r(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_p$ (solid curves). The geometry of their intersections highlights the following principles:

- For $0 < p \leq 1$, the ℓ_p ball has non-smooth sharp "corners" aligned with the coordinate axes. Consequently, the risk contours often intersect the feasible region at these corners, leading to sparse solutions. In contrast, for $p = 2$ the ball is smooth and rounded, so sparsity is not promoted.
- As p decreases toward 0, the corners become sharper and the sparsity-inducing effect stronger. For instance, in the right panel, the ℓ_1 penalty does not produce a sparse minimizer, whereas the non-convex $\ell_{0.4}$ penalty does.
- Geometrically, sparsity emerges precisely from these non-smooth axis-aligned singularities. When $p > 1$, the ℓ_p ball lacks such singularities, so the minimizer is unlikely to lie on a coordinate axis and sparsity is no longer encouraged.

Balancing computational tractability with the ability to promote sparsity, the Lasso (9) arises as the natural choice: it can be interpreted as a convex relaxation of the ℓ_0 problem (10). This naturally leads to the following question: under what conditions can ℓ_1 regularization exactly recover the solution of the ℓ_0 problem?

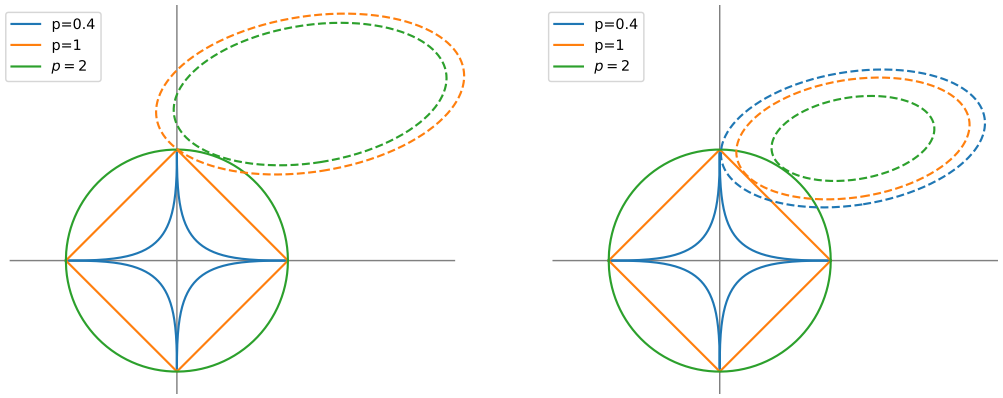

Figure 2: Geometric intuition of ℓ_p regularization. The solid curves represent ℓ_p balls for different values of p, and the dashed curves represent contours of the empirical risk. **Left:** Both ℓ_1 and $\ell_{0.4}$ solutions lie on the coordinate axes, yielding sparse estimates. **Right:** The ℓ_1 penalty fails to produce a sparse solution, whereas the sharper corners of the $\ell_{0.4}$ ball still enforce sparsity. These illustrations highlight why ℓ_p regularization with $p \leq 1$ preserves the sparsity-inducing effect of the ℓ_0 norm, while ℓ_p with $p > 1$ does not.

1.3.1 Compressed sensing

Up to this point, our discussion of ℓ_1 regularization has focused on the Lasso for linear regression with noisy data. A rigorous analysis of when Lasso can exactly recover the true sparse parameter $\boldsymbol{\beta}^*$ is more involved and requires additional machinery.

To convey the key idea in a simpler setting, we turn to the closely related problem of *compressed sensing*, which studies the recovery of sparse solutions to underdetermined linear systems. Although the statistical formulations are not identical, the central message is the same: under appropriate structural conditions on

X, the intractable ℓ_0 problem can be replaced by ℓ_1 minimization without losing sparsity. The compressed sensing framework provides a clean and intuitive theorem (Candès–Tao, 2006) that illustrates this principle, and thus serves as an accessible proxy for understanding why ℓ_1 regularization promotes sparsity.

Mathematically, compressed sensing begins with an underdetermined linear system

$$X\boldsymbol{\beta} = \mathbf{y}, \qquad X \in \mathbb{R}^{n\times d},\ d \gg n.$$

Since the number of equations is much smaller than the number of unknowns, such a system typically admits infinitely many solutions. A natural way to single out a meaningful solution is to prefer the *sparsest* one, i.e., the solution with the fewest nonzero entries. This leads to the ℓ_0 minimization problem

$$\begin{aligned} \min \quad & \|\boldsymbol{\beta}\|_0, \\ \text{s.t.} \quad & X\boldsymbol{\beta} = \mathbf{y}. \end{aligned} \tag{13}$$

The central question is:

can the sparsest solution be recovered efficiently?

Since direct ℓ_0 minimization is combinatorial and generally intractable, one seeks a tractable alternative. A key insight from compressed sensing is that, under suitable conditions on X, the ℓ_0 problem can be replaced by its convex surrogate – the ℓ_1 minimization problem – while still yielding the same sparse solution:

$$\begin{aligned} \min \quad & \|\boldsymbol{\beta}\|_1, \\ \text{s.t.} \quad & X\boldsymbol{\beta} = \mathbf{y}. \end{aligned} \tag{14}$$

Definition 1.7. A vector $\boldsymbol{\beta} \in \mathbb{R}^d$ is called *s-sparse* if $\|\boldsymbol{\beta}\|_0 \le s$, where s is a positive integer referred to as the sparsity level.

A central condition is the *restricted isometry property (RIP)*, which requires the matrix X to act almost like an isometry when restricted to sparse vectors:

Definition 1.8 (Restricted Isometry Property (RIP))**.** The matrix X satisfies the RIP of order s with constant $\delta_s \in (0,1)$ if it holds for all s-sparse vectors $\boldsymbol{\beta}$ that

$$(1-\delta_s)\|\boldsymbol{\beta}\|_2^2 \ \le\ \|X\boldsymbol{\beta}\|_2^2 \ \le\ (1+\delta_s)\|\boldsymbol{\beta}\|_2^2$$

Intuitively, RIP guarantees that different sparse vectors remain well separated after multiplication by X: no two distinct sparse signals are mapped too close to each other.

Theorem 1.9 (Candès–Tao, 2006)**.** *Let $\boldsymbol{\beta}_0$ be a solution of the ℓ_0 problem (13), and let $\boldsymbol{\beta}_1$ be a solution of the ℓ_1 problem (14). If X satisfies the RIP of order $2s$ with constant $\delta_{2s} < \sqrt{2}-1$, then*

$$\|\boldsymbol{\beta}_1 - \boldsymbol{\beta}_0\|_1 \ \le\ C_0\, \|\boldsymbol{\beta}_0 - T_s(\boldsymbol{\beta}_0)\|_1,$$

where $T_s(\boldsymbol{\beta})$ denotes the best s-sparse approximation of $\boldsymbol{\beta}$, obtained by retaining its s largest entries in magnitude and setting the rest to zero. In particular, if $\boldsymbol{\beta}_0$ is exactly s-sparse, then $\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0$.

Example 1.10 (Best s-sparse approximation)**.** *Let $\boldsymbol{\beta} = (3.2,\ 0.1,\ 0,\ -4.5,\ 2.7)^\top$ and take $s = 2$. Then,*

$$T_2(\boldsymbol{\beta}) = (3.2,\ 0,\ 0,\ -4.5,\ 0)^\top.$$

Theorem 1.9 should not only be read as an abstract statement, but as a description of the underlying phenomenon. It says: if the measurement matrix X preserves the geometry of sparse vectors (RIP), then solving the tractable ℓ_1 problem recovers essentially the same solution as the intractable ℓ_0 problem.

- If the true solution $\boldsymbol{\beta}_0$ is exactly s-sparse, then ℓ_1 recovers it *exactly*.
- If $\boldsymbol{\beta}_0$ is not perfectly sparse but is *compressible* (most energy concentrated in a few coordinates), then ℓ_1 still recovers it *approximately*, with the error controlled by the small entries discarded by $T_s(\boldsymbol{\beta}_0)$.

The proof is technical and can be found in [Candes et al., 2006].

Remark 1.11 (Compressed Sensing vs. Lasso). It is worth emphasizing the difference between compressed sensing and the Lasso in machine learning. In compressed sensing, the design matrix X can often be chosen or engineered (e.g., by taking random Gaussian or Fourier measurements), and the RIP condition provides a guideline for guaranteeing exact recovery. In contrast, in the Lasso for regression, the matrix X is the data matrix determined by the problem at hand and is generally not under our control. As a result, X may not satisfy RIP. Nevertheless, in practice the Lasso still performs remarkably well in promoting sparsity and yielding accurate predictions, even beyond the strict RIP setting.

1.3.2 A generalization analyses of Lasso

Consider the situation where the ground truth $\boldsymbol{\beta}^*$ is sparse and the signal $\mathbf{y}$ is contaminated by noise:

$$y_i = {\boldsymbol{\beta}^*}^\top \mathbf{x}_i + \varepsilon_i. \tag{15}$$

Let $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)^\top \in \mathbb{R}^n$ be the noise vector. Denote by $\hat{\boldsymbol{\beta}}$ the Lasso estimator produced by (9).

Proposition 1.12. *For the Lasso estimator, if* $\lambda \geq \lambda_n := \frac{\|X^\top \boldsymbol{\varepsilon}\|_\infty}{n}$, *then*

$$\frac{1}{2n}\|X(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)\|_2^2 \leq 2\lambda \|\boldsymbol{\beta}^*\|_1. \tag{16}$$

This proposition shows that the empirical risk is well-controlled when the penalization is relatively large with respect to the noise, i.e. $\lambda \geq \|X^\top \boldsymbol{\varepsilon}\|_\infty / n$. The proof given below is simple but very representative for generalization analysis of regularized estimators. The main idea is to compare the estimator of interest with "ground truth" and to identify some conditions such that the estimator has properties similar to ground truth. This comparison trick will appear many times in this book and we will see in exercise that this comparison can also lead to the controlledness of the norm of Lasso estimator.

Proof. By comparing $\hat{\boldsymbol{\beta}}$ with the ground truth $\boldsymbol{\beta}^*$, we have

$$\frac{1}{2n}\|X\hat{\boldsymbol{\beta}} - \mathbf{y}\|_2^2 + \lambda\|\hat{\boldsymbol{\beta}}\|_1 \leq \frac{1}{2n}\|X\boldsymbol{\beta}^* - \mathbf{y}\|_2^2 + \lambda\|\boldsymbol{\beta}^*\|_1 \tag{17}$$

Notice that $\mathbf{y} = X\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}$ implies for any $\boldsymbol{\beta} \in \mathbb{R}^d$ that

$$\|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 = \|X\boldsymbol{\beta} - X\boldsymbol{\beta}^*\|_2^2 + 2\langle X(\boldsymbol{\beta} - \boldsymbol{\beta}^*), \boldsymbol{\varepsilon}\rangle + \|\boldsymbol{\varepsilon}\|_2^2. \tag{18}$$

Inserting (18) into the comparison inequality (17), we obtain the following estimates.

$$\frac{1}{2n}\|X(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)\|_2 \leq \frac{1}{n}\left\langle X(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*), \boldsymbol{\varepsilon}\right\rangle + \lambda\|\boldsymbol{\beta}^*\|_1 - \lambda\|\hat{\boldsymbol{\beta}}\|_1$$

$$\leq \frac{\|X^\top \boldsymbol{\varepsilon}\|_\infty}{n}\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_1 + \lambda(\|\boldsymbol{\beta}^*\|_1 - \|\hat{\boldsymbol{\beta}}\|_1).$$

Since $\lambda \geq \frac{\|X^\top \boldsymbol{\varepsilon}\|_\infty}{n}$, we have

$$\begin{aligned}\frac{1}{2n}\|X(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)\|_2^2 &\leq \lambda\left(\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_1 - \|\hat{\boldsymbol{\beta}}\|_1 + \|\boldsymbol{\beta}^*\|_1\right)\\ &\leq 2\lambda\|\boldsymbol{\beta}^*\|_1.\end{aligned}$$

□

Theorem 1.13. *Let X_j is the j-th column of X for $j \in [d]$. Assume $\|X_j\|_2^2 = n, \forall j \in [d]$ and the noise $\varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$ for $i = 1, \ldots, n$. For any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the sampling of the random noise, we have*

$$\lambda_n \leq C\sigma\sqrt{\frac{\log(d/\delta)}{n}},$$

and if $\lambda \asymp \lambda_n$, we have

$$\frac{1}{n}\|X(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)\|_2^2 \leq \frac{C\sigma\sqrt{\log(d/\delta)}\|\boldsymbol{\beta}^*\|_1}{\sqrt{n}}.$$

This theorem suggests that the regularization hyperparameter λ should decrease as the sample size n increases, which aligns with intuition: weaker regularization is needed when more data are available. Moreover, if the ground-truth parameter $\boldsymbol{\beta}^*$ is s-sparse, then $\|\boldsymbol{\beta}^*\|_1 = O(s)$, so the error depends linearly on the sparsity level s but only *logarithmically* on the ambient dimension d. This explains why ℓ_1 regularization is particularly advantageous in high-dimensional settings when the underlying signal is sparse.

Proof. By Proposition 1.12, what remains is to estimate $\|X^\top \boldsymbol{\varepsilon}\|_\infty = \max_{j\in[d]} |X_j^\top \boldsymbol{\varepsilon}|$. Since $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I_d)$, $X_j^\top \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \|X_j\|_2^2\, \sigma^2)$. Then, we have

$$\begin{aligned}\mathbb{P}\left\{\max_{1\leq j\leq d}\left|X_j^\top \boldsymbol{\varepsilon}\right| \geq t\right\} &\leq d\,\mathbb{P}\left\{\left|X_1^\top \boldsymbol{\varepsilon}\right| \geq t\right\}\\ &= 2d\int_t^\infty \frac{1}{\sqrt{2\pi\sigma^2 n}}\, e^{-\frac{z^2}{2\sigma^2 n}}\,\mathrm{d}z\\ &= 2d\frac{1}{\sqrt{2\pi}}\int_{\frac{t}{\sigma\sqrt{n}}}^\infty e^{-\frac{u^2}{2}}\,\mathrm{d}u.\end{aligned}$$

Notice that the tail of standard norm distribution satisfies

$$\frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{z^2}{2}}\,\mathrm{d}z \leq e^{-\frac{x^2}{2}}, \qquad \forall x > 0.$$

Therefore,

$$\mathbb{P}\left\{\max_{1\leq j\leq d}\left|X_j^\top \boldsymbol{\varepsilon}\right| \geq t\right\} \leq \frac{2d}{\sqrt{2\pi}}e^{-\frac{t^2}{2\sigma^2 n}}.$$

Let the failure probability $\frac{2d}{\sqrt{2\pi}}e^{-\frac{t^2}{2\sigma^2 n}} \leq \delta$. We have $t \geq C\sigma\sqrt{n\log(d/\delta)}$. Therefore, we can conclude that with probability at least $1 - \delta$, it holds that

$$\frac{\|X^\top \boldsymbol{\varepsilon}\|_\infty}{n} \leq C\sigma\sqrt{\frac{\log(d/\delta)}{n}}.$$

Then, we complete the proof by Proposition 1.12. □

The normalization condition $\|X_j\|_2^2 = \sum_{i=1}^n x_{i,j}^2 = n$ ensures that the value of each coordinate is roughly $O(1)$. From the proof, one can see the Gaussian assumption of noise: $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ is not essential and it can be replaced by any distribution as long as its tail decays like a Gaussian, i.e., $\mathbb{P}\{|\varepsilon_i| > t\} \leq C_1 e^{-C_2 t^2}$ for some positive constants C_1, C_2. This kind of tail property is often referred to as being sub-Gaussian (see [Vershynin, 2018, Section 2] for more details).

Exercise 1.14. If $\lambda \geq \lambda_n$, then we have $\|\hat{\boldsymbol{\beta}}\|_1 \leq 3\|\boldsymbol{\beta}^*\|_1$.

Hence, when $\lambda \geq \lambda_n$, we have 1) the empirical risk is small; 2) the parameter norm of $\hat{\boldsymbol{\beta}}$ is well-controlled. These two properties together, as we will see later, can guarantee the good generalization.

1.3.3 Variable Selection

A distinctive feature of the Lasso is its ability to perform variable selection in addition to prediction. Thanks to the sparsity-promoting effect of ℓ_1 regularization, Lasso can shrink the coefficients of unimportant variables exactly to zero (see Theorem 1.9). The variables with nonzero coefficients naturally account for the prediction. By contrast, ridge regression does not share this property.

To better understand this selection mechanism, it is instructive to examine the *regularization path*, which refers to the trajectory of the estimator as the penalty parameter λ varies:

$$\hat{\boldsymbol{\beta}}(\lambda) = \operatorname*{argmin}_{\boldsymbol{\beta}} \frac{1}{2n}\|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \qquad \lambda \geq 0.$$

This concept applies to any regularized estimator, though the resulting paths differ across methods.

Figure 3 illustrates this comparison by plotting the regularization paths of both the Lasso and ridge regression (Ridge). For the Lasso, when λ is small, many coefficients are nonzero. As λ increases, coefficients corresponding to less important variables are successively shrunk to exactly zero and stay there, while only the most influential variables remain active at large λ. This sequential elimination underlies the Lasso's ability to perform variable selection. In contrast, under Ridge all coefficients decrease smoothly as λ grows, but none of them ever reaches zero. When λ becomes large, the coefficients are all shrunken to small values of roughly the same magnitude, making it impossible to distinguish important variables from unimportant ones. This illustrates why Ridge cannot be used for variable selection.

1.3.4 Algorithms for Lasso

Let us begin with the one-dimensional case:

$$S_\lambda(x) = \operatorname*{argmin}_{t \in \mathbb{R}} \frac{1}{2}(t - x)^2 + \lambda|t|. \tag{19}$$

The minimizer admits a closed-form expression:

$$S_\lambda(x) = \begin{cases} x - \lambda, & x > \lambda, \\ 0, & -\lambda \leq x \leq \lambda, \\ x + \lambda, & x < -\lambda. \end{cases} \tag{20}$$

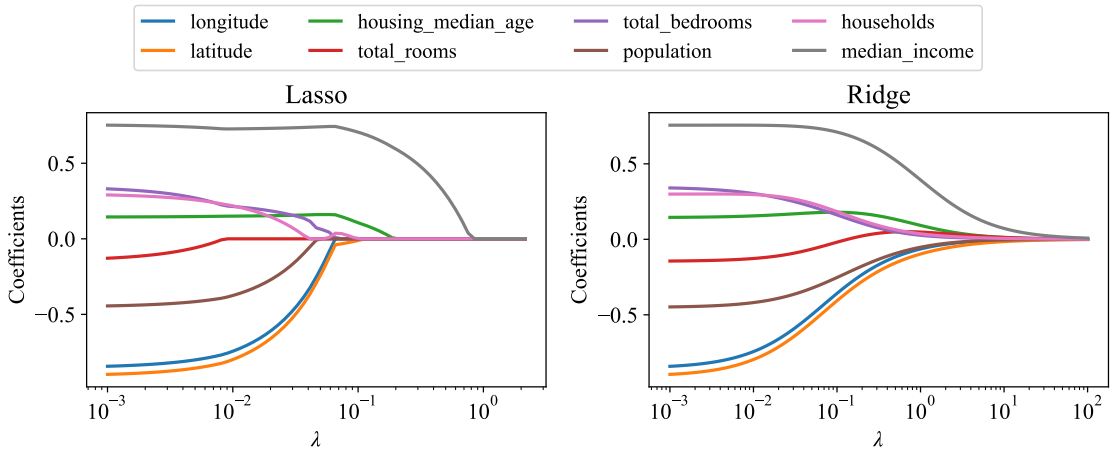

Figure 3: Regulation paths $\lambda \mapsto \hat{\beta}(\lambda)$ for Lasso (**Left**) and Ridge (**Right**). The response is the median house value, the predictors are median income, average house age, average number of rooms, population, and others. The data come from the California housing dataset, which is available in `sklearn`.

The function $S_\lambda(\cdot)$ is known as the *soft-thresholding operator*. By comparison, the *hard-thresholding operator* is defined as

$$H_\lambda(x) = \operatorname*{argmin}_{t \in \mathbb{R}} \frac{1}{2}(t-x)^2 + \frac{\lambda^2}{2}|t|_0, \tag{21}$$

which also has a closed-form solution:

$$H_\lambda(x) = \begin{cases} x, & |x| > \lambda, \\ 0, & |x| \le \lambda. \end{cases} \tag{22}$$

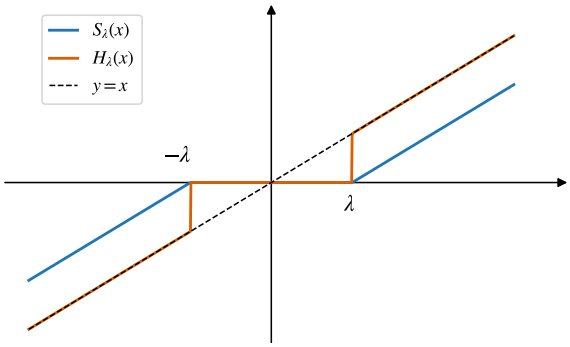

Figure 4: Comparison of the soft-thresholding operator S_λ and the hard-thresholding operator H_λ, together with the identity line $y = x$.

Figure 4 compares the two operators. Both set small coefficients to zero, but they behave differently on large coefficients.

- **Soft thresholding** (S_λ) shrinks all coefficients toward zero by λ and zeroes them out when $|x| \le \lambda$. It is continuous but introduces shrinkage bias.

- **Hard thresholding** (H_λ) keeps large coefficients unchanged and sets small ones to zero whenever $|x| \le \lambda$. It avoids shrinkage bias but is discontinuous at the threshold.

In short, S_λ gives biased yet smooth shrinkage, while H_λ gives unbiased but discontinuous selection. In contrast, the Ridge solution

$$R_\lambda(x) \;:=\; \operatorname*{argmin}_{t\in\mathbb{R}} \frac{1}{2}(t-x)^2 + \frac{\lambda}{2}|t|^2 = \frac{x}{1+\lambda},$$

never drives coefficients to exactly zero. Hence Ridge shrinks parameters but does not promote sparsity.

For the d-dimensional case with $d > 1$, the problem can be decomposed into a sequence of one-dimensional subproblems using the *coordinate descent* method. Given the objective $f(x_1, \dots, x_d)$, one iteration updates

$$(x_1^{(t)}, x_2^{(t)}, \dots, x_d^{(t)}) \;\mapsto\; (x_1^{(t+1)}, x_2^{(t+1)}, \dots, x_d^{(t+1)})$$

by cycling through the coordinates:

$$\begin{aligned}
x_1^{(t+1)} &\in \operatorname*{argmin}_{x_1\in\mathbb{R}} f(x_1, x_2^{(t)}, \dots, x_d^{(t)}),\\
x_2^{(t+1)} &\in \operatorname*{argmin}_{x_2\in\mathbb{R}} f(x_1^{(t+1)}, x_2, \dots, x_d^{(t)}),\\
&\;\;\vdots\\
x_d^{(t+1)} &\in \operatorname*{argmin}_{x_d\in\mathbb{R}} f(x_1^{(t+1)}, \dots, x_{d-1}^{(t+1)}, x_d).
\end{aligned}$$

This corresponds to a **Gauss–Seidel update**, where each coordinate uses the latest values of the preceding variables. In contrast, the **Jacobi update** keeps all coordinates fixed at their old values when solving the subproblems:

$$\begin{aligned}
x_1^{(t+1)} &\in \operatorname*{argmin}_{x_1\in\mathbb{R}} f(x_1, x_2^{(t)}, \dots, x_d^{(t)}),\\
x_2^{(t+1)} &\in \operatorname*{argmin}_{x_2\in\mathbb{R}} f(x_1^{(t)}, x_2, \dots, x_d^{(t)}),\\
&\;\;\vdots\\
x_d^{(t+1)} &\in \operatorname*{argmin}_{x_d\in\mathbb{R}} f(x_1^{(t)}, \dots, x_{d-1}^{(t)}, x_d).
\end{aligned}$$

Remark 1.15. The Gauss–Seidel update often converges faster in practice because it immediately exploits the latest information, but it is inherently sequential. The Jacobi update, by contrast, updates all coordinates in parallel using only the old iterate, making it more suitable for distributed or parallel computation, albeit sometimes with slower convergence in terms of number of iterations.

Next we update a single coordinate while keeping all others fixed. Let $X_k \in \mathbb{R}^n$ denote the k-th column of X and define the partial residual $\tilde{\mathbf{y}}_j \;=\; \mathbf{y} - \sum_{k\neq j} X_k\beta_k$. Then

$$\|X\boldsymbol{\beta} - \mathbf{y}\|_2^2 \;=\; \|X_j\beta_j - \tilde{\mathbf{y}}_j\|_2^2 = \|\tilde{\mathbf{y}}_j\|_2^2 \;-\; 2\,\beta_j\langle X_j, \tilde{\mathbf{y}}_j\rangle \;+\; \|X_j\|_2^2\,\beta_j^2.$$

Hence the j-th coordinate subproblem for Lasso is

$$\begin{aligned}\operatorname*{argmin}_{\beta_j\in\mathbb{R}} \frac{1}{2n}\|X\boldsymbol{\beta}-\mathbf{y}\|_2^2+\lambda\|\boldsymbol{\beta}\|_1 &= \operatorname*{argmin}_{\beta_j\in\mathbb{R}} \frac{1}{2n}\big(\|X_j\|_2^2\beta_j^2-2\langle X_j,\tilde{\mathbf{y}}_j\rangle\,\beta_j\big)+\lambda|\beta_j| \\ &\equiv \operatorname*{argmin}_{\beta_j\in\mathbb{R}} \frac{1}{2}\left(\beta_j-\frac{\langle X_j,\tilde{\mathbf{y}}_j\rangle}{\|X_j\|_2^2}\right)^2+\frac{n\lambda}{\|X_j\|_2^2}\,|\beta_j|,\end{aligned}$$

where $\equiv$ uses multiplication by the positive constant $n/\|X_j\|_2^2$ and drops β_j–independent terms. By (19), the one–coordinate update is given by

$$\beta_j \;\leftarrow\; S_{\frac{n\lambda}{\|X_j\|_2^2}}\left(\frac{\langle X_j,\tilde{\mathbf{y}}_j\rangle}{\|X_j\|_2^2}\right). \tag{23}$$

Cycle (23) over $j=1,2,\ldots,d$ (Gauss–Seidel or Jacobi) until convergence.

If the columns are standardized so that $\|X_j\|_2^2=n$, then

$$\beta_j \leftarrow S_\lambda\left(\frac{1}{n}\,\langle X_j,\tilde{\mathbf{y}}_j\rangle\right),$$

i.e., the threshold is exactly λ and the "least-squares target" is the (partial) correlation $\langle X_j,\tilde{\mathbf{y}}_j\rangle/n$.

The coordinate descent method is simple and easy to implement. In practice, however, the most widely used approaches for solving large-scale ℓ_1-regularized problems are the alternating direction method of multipliers (ADMM) and its variants; see [Boyd et al., 2011] for details. That said, coordinate updates remain a very general design principle and can be applied in many other optimization settings beyond solving the Lasso problem.

2 Kernel methods

2.1 From Features to Kernels

The preceding methods apply only to represent linear functions. A natural extension to represent nonlinear functions is to consider

$$f(\mathbf{x};\boldsymbol{\beta})=\sum_{j=1}^{m}\beta_j\,\phi_j(\mathbf{x}), \tag{24}$$

where $\{\phi_j\}_{j=1}^m$ are nonlinear basis functions, such as Fourier modes or polynomials. Although the model remains linear in the parameters $\boldsymbol{\beta}$, the induced function class is nonlinear in $\mathbf{x}$.

The basis functions are often called "features" in machine learning and the corresponding feature map $\phi:\mathcal{X}\to\mathbb{R}^m$ is given by

$$\phi(\mathbf{x})=(\phi_1(\mathbf{x}),\ldots,\phi_m(\mathbf{x}))^\top\in\mathbb{R}^m,$$

where $\mathcal{X}$ is the input space and $\mathbb{R}^m$ is the feature space. In many applications, one often chooses $m>d$, where the feature map lifts the input $\mathbf{x}$ to a higher-dimensional feature space.

To fit the model (24), consider the ridge regression in feature space:

$$\hat{\boldsymbol{\beta}}=\operatorname*{argmin}_{\boldsymbol{\beta}}\frac{1}{2n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m}\beta_j\phi_j(\mathbf{x}_i)-y_i\right)^2+\frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2$$

$$= \operatorname*{argmin}_{\boldsymbol{\beta}} \frac{1}{2n}\|\hat{\Phi}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2, \tag{25}$$

where $\hat{\Phi} := \big(\phi_j(\mathbf{x}_i)\big)_{1\le i\le n, 1\le j\le m} \in \mathbb{R}^{n\times m}$ is the feature matrix. Then,

$$\hat{\boldsymbol{\beta}} = \left(\tfrac{1}{n}\hat{\Phi}^\top\hat{\Phi} + \lambda I_m\right)^{-1} \tfrac{1}{n}\hat{\Phi}^\top\mathbf{y}. \tag{26}$$

The function represented by $\hat{\boldsymbol{\beta}}$ is given by $f(\mathbf{x};\hat{\boldsymbol{\beta}}) = \sum_{j=1}^m \hat{\beta}_j\phi_j(\mathbf{x})$. The following theorem shows that $\hat{f}$ can be written as a linear combination of

$$k(\mathbf{x},\mathbf{x}') := \phi(\mathbf{x})^\top\phi(\mathbf{x}') = \sum_{j=1}^m \phi_j(\mathbf{x})\phi_j(\mathbf{x}'). \tag{27}$$

Here, $k : \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ is called the kernel function, and the rigorous definition is given later. Specifically, we have the following representation theorem for the solution (25).

Theorem 2.1. *Let $K = (k(\mathbf{x}_i,\mathbf{x}_j)) \in \mathbb{R}^{n\times n}$ be the Gram matrix. For any $\lambda > 0$, let $\hat{\boldsymbol{\alpha}} = \left(\frac{1}{n}K + \lambda I_n\right)^{-1}\frac{1}{n}\mathbf{y} \in \mathbb{R}^n$. Then the solution of (25) can be rewritten as*

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^n \hat{\alpha}_i\phi(\mathbf{x}_i), \qquad f(\mathbf{x};\hat{\boldsymbol{\beta}}) = \sum_{i=1}^n \hat{\alpha}_i k(\mathbf{x}_i,\mathbf{x}). \tag{28}$$

Proof. For any $A\in\mathbb{R}^{n\times m}$ and $\lambda > 0$, we have the following identity

$$(A^\top A + I_m)^{-1}A^\top = A^\top(AA^\top + I_n)^{-1}.$$

Substituting $A = \frac{1}{\sqrt{n}}\hat{\Phi}$ yields

$$\hat{\boldsymbol{\beta}} = \left(\tfrac{1}{n}\hat{\Phi}^\top\hat{\Phi} + \lambda I_m\right)^{-1}\tfrac{1}{n}\hat{\Phi}^\top\mathbf{y} = \hat{\Phi}^\top\left(\tfrac{1}{n}\hat{\Phi}\hat{\Phi}^\top + \lambda I_n\right)^{-1}\tfrac{1}{n}\mathbf{y} = \hat{\Phi}^\top\hat{\boldsymbol{\alpha}}$$

and moreover,

$$\begin{aligned} f(\mathbf{x};\hat{\boldsymbol{\beta}}) &= \sum_{j=1}^m \phi_j(\mathbf{x})\hat{\beta}_j = \sum_{j=1}^m \phi_j(\mathbf{x})\sum_{i=1}^n \phi_j(\mathbf{x}_i)\hat{\alpha}_i \\ &= \sum_{i=1}^n \hat{\alpha}_i\left(\sum_{j=1}^m \phi_j(\mathbf{x}_i)\phi_j(\mathbf{x})\right) = \sum_{i=1}^n \hat{\alpha}_i k(\mathbf{x}_i,\mathbf{x}). \end{aligned}$$

□

Theorem 2.1 leads to two important implications:

- **Computational trade-off.** In (26), the dominant cost comes from inverting an $m\times m$ matrix, whereas in (28) it reduces to an $n\times n$ inversion. Hence, (28) is computationally more efficient whenever $m > n$, i.e., when the feature dimension exceeds the sample size.
- **Kernel trick.** Any linear algorithm whose computations depend on data only through inner products $\langle\mathbf{x}_i,\mathbf{x}_j\rangle$ can be made nonlinear by substituting $\langle\mathbf{x}_i,\mathbf{x}_j\rangle \mapsto k(\mathbf{x}_i,\mathbf{x}_j)$. In practice, this amounts to replacing $X^\top X$ (or $XX^\top$) by the kernel matrix K. The resulting method operates in the feature space $\mathcal{H}$ without ever computing $\phi(x)$.

This observation motivates the formal notion of *kernels* and kernel methods, which we introduce next.

2.2 Feature Maps, Kernels, and Positive Semidefiniteness

In the previous discussion, we considered models constructed from a finite collection of basis functions and carried out elementary calculations in this finite-dimensional setting to illustrate how kernels arise. A natural question is whether one can move beyond such finite representations to more general, possibly infinite-dimensional feature spaces. Kernel methods provide exactly such a framework. The central observation, formalized in Theorem 2.1, is that the feature vectors themselves are never needed explicitly; instead, only their pairwise inner products are required.

Formally, a *feature map* is a mapping

$$\phi : \mathcal{X} \to \mathcal{H},$$

where $\mathcal{X}$ is the input space and $\mathcal{H}$ is a Hilbert space, which may be infinite-dimensional. The Hilbert space $\mathcal{H}$ is referred to as the *feature space*. The purpose of the feature map is to transfer the learning problem from the input space $\mathcal{X}$ to the feature space $\mathcal{H}$. Algorithms that originally act on raw inputs $\mathbf{x} \in \mathcal{X}$ can equivalently operate on their images $\phi(\mathbf{x}) \in \mathcal{H}$. This often has the effect of transforming nonlinear relations in $\mathcal{X}$ into linear ones in $\mathcal{H}$, where linear methods and inner products can be applied effectively. As an example, in (24) the feature map takes values in $\mathbb{R}^m$ and is given by

$$\phi(\mathbf{x}) = \big(\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_m(\mathbf{x})\big)^\top \in \mathbb{R}^m,$$

where $\phi_1, \ldots, \phi_m$ are basis functions.

Definition 2.2 (Kernel function)**.** A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *kernel* if there exists a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that

$$k(\mathbf{x}, \mathbf{x}') \;=\; \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}.$$

Throughout this section, $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ denote the inner product and norm in the feature space $\mathcal{H}$. For notational simplicity, we omit the explicit subscript $\mathcal{H}$ whenever no confusion arises. One can view the kernel function as measuring the similarity between two inputs $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ by using the inner product of their feature representations, $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$.

The definition above is not directly verifiable in practice, since the feature map ϕ is typically unknown. It is therefore useful to introduce an intrinsic characterization of kernels in terms of positive semidefiniteness.

Definition 2.3 (PSD function)**.** A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called *positive semidefinite (PSD)* if

- $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ (symmetry);
- For any $n \in \mathbb{N}$, $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, and $\boldsymbol{\alpha} \in \mathbb{R}^n$, it holds $\sum_{i,j=1}^n \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \;\geq\; 0$.

Let $K = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^n \in \mathbb{R}^{n \times n}$ denote the kernel (also called the Gram matrix). The requirement that k is PSD means precisely that K is positive semidefinite for any choice of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

It is easy to verify that a kernel k is always PSD, as

$$\sum_{i,j=1}^n \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{i,j=1}^n \alpha_i \alpha_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \left\| \sum_{i=1}^n \alpha_i \phi(\mathbf{x}_i) \right\|^2 \;\geq\; 0.$$

The following theorem shows that PSD is not only necessary for k to be a kernel but also sufficient.

Theorem 2.4 (Moore–Aronszajn [Aronszajn, 1950])**.** *For any PSD function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that*

$$k(\mathbf{x}, \mathbf{x}') \;=\; \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}.$$

A detailed proof can be found in [Steinwart and Christmann, 2008, Theorem 4.16]. This result shows that kernels and PSD functions are equivalent: to verify that k is a valid kernel, it suffices to check its positive semidefiniteness.

2.3 Examples of kernels

Here, we provide a list of popular kernels.

Polynomial kernel. The polynomial kernel is defined as

$$k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\top \mathbf{x}')^s,$$

which corresponds to embedding the input into a feature space spanned by all monomials of the input variables up to degree s.

- *Linear case ($s = 1$).* In this case, $k(\mathbf{x}, \mathbf{x}') = 1 + \mathbf{x}^\top \mathbf{x}' = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$, where the feature map is

$$\phi(\mathbf{x}) = (1, x_1, \ldots, x_d)^\top.$$

- *Quadratic case ($s = 2$).* The feature map consists of constant, linear terms, cross terms, and quadratic terms:

$$\phi(\mathbf{x}) = \big(\underbrace{1}_{\text{constant}}, \underbrace{\sqrt{2}x_1, \ldots, \sqrt{2}x_d}_{\text{linear terms}}, \underbrace{\sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{d-1}x_d}_{\text{cross terms}}, \underbrace{x_1^2, \ldots, x_d^2}_{\text{quadratic terms}}, \big)^\top.$$

  One verifies directly that

$$\begin{aligned}
\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle &= 1 + 2\sum_{i=1}^{d} x_i x_i' + 2\sum_{i<j} (x_i x_j)(x_i' x_j') + \sum_{i=1}^{d} x_i^2 (x_i')^2 \\
&= 1 + 2\sum_{i=1}^{d} x_i x_i' + \Big(\sum_{i=1}^{d} x_i x_i' \Big)^2 \\
&= \big(1 + \mathbf{x}^\top \mathbf{x}'\big)^2.
\end{aligned}$$

Gaussian kernel. The Gaussian kernel is defined by

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2} \right),$$

where $\sigma > 0$ is a scale parameter, often referred to as the *bandwidth*. The bandwidth controls the notion of similarity: if $\|\mathbf{x} - \mathbf{x}'\| \ll \sigma$, then $k(\mathbf{x}, \mathbf{x}') \approx 1$, meaning the two points are regarded as highly similar; if $\|\mathbf{x} - \mathbf{x}'\| \gg \sigma$, then $k(\mathbf{x}, \mathbf{x}') \approx 0$, meaning the two points are essentially unrelated. Geometrically, a small

σ yields a sharply peaked kernel that emphasizes local neighborhoods, while a large σ produces a broad kernel that enforces global smoothness. Thus, σ governs the balance between local sensitivity and global averaging in learning with Gaussian kernels.

To see the feature map explicitly, consider the one-dimensional case ($d = 1$) with $\sigma = 1$:

$$k(x, x') = e^{-\frac{x^2}{2} - \frac{x'^2}{2}} e^{xx'} = e^{-\frac{x^2}{2} - \frac{x'^2}{2}} \sum_{n=0}^{\infty} \frac{1}{n!} x^n x'^n = \langle \phi(x), \phi(x') \rangle,$$

where the corresponding feature map $\phi : \mathbb{R} \to \ell^2$ is

$$\phi(x) = e^{-\frac{x^2}{2}} \left(1,\ x,\ \tfrac{1}{\sqrt{2!}} x^2,\ \ldots,\ \tfrac{1}{\sqrt{n!}} x^n,\ \ldots\right) \in \ell^2.$$

Thus, in this case the Gaussian kernel corresponds to an infinite-dimensional feature space ℓ^2 with features given by all monomials, scaled by the Gaussian weight $e^{-x^2/2}$.

Laplace kernel. The Laplace kernel is defined as

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2}{\sigma}\right),$$

where $\sigma > 0$ is a scale parameter. Compared with the Gaussian kernel, the Laplace kernel decays more slowly but is less smooth, since it is only Lipschitz continuous rather than infinitely differentiable. Recent work has revealed close connections between the Laplace kernel and neural network models in the kernel regime; see [Chen and Xu, 2020, Geifman et al., 2020].

Dot-product kernels [2]**.** Let $\mathbb{S}^{d-1} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 = 1\}$ denote the unit sphere in $\mathbb{R}^d$. A kernel $k : \mathbb{S}^{d-1} \times \mathbb{S}^{d-1} \to \mathbb{R}$ is called a *dot-product kernel* if there exists a function $\kappa : [-1, 1] \to \mathbb{R}$ such that

$$k(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}^\top \mathbf{x}'), \qquad \mathbf{x}, \mathbf{x}' \in \mathbb{S}^{d-1}.$$

In other words, the kernel value depends only on the inner product between the two inputs.

As an example, the Gaussian kernel with $\sigma = 1$ restricted to the sphere is a dot-product kernel:

$$k(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2}} = e^{-1 + \mathbf{x}^\top \mathbf{x}'} = \kappa(\mathbf{x}^\top \mathbf{x}'),$$

where $\kappa(t) = e^{t-1}$, which is analytic on the entire domain $[-1, 1]$.

Similarly, the Laplace kernel with $\sigma = 1$ restricted to the sphere also admits a dot-product form:

$$k(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x} - \mathbf{x}'\|_2} = e^{-\sqrt{2 - 2\mathbf{x}^\top \mathbf{x}'}} = \kappa(\mathbf{x}^\top \mathbf{x}'),$$

with $\kappa(t) = e^{-\sqrt{2-2t}}$. Unlike the Gaussian case, here κ is not differentiable at $t = 1$, reflecting the reduced smoothness of the Laplace kernel.

Exercise 2.5. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be an activation function and let $\pi_0 = \mathrm{Unif}(\mathbb{S}^{d-1})$ denote the uniform distribution on the unit sphere. For a given $\mathbf{u} \in \mathbb{R}^d$, the function $\mathbf{x} \mapsto \sigma(\mathbf{u}^\top \mathbf{x})$ is called a *neuron*. Define the kernel

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{u} \sim \pi_0} \left[\sigma(\mathbf{u}^\top \mathbf{x})\, \sigma(\mathbf{u}^\top \mathbf{x}')\right].$$

Show that k, when restricted to the unit sphere, is a dot-product kernel.

[2] Also referred to as inner-product kernels.

Kernel calculus. New kernels can be systematically constructed from existing ones. Let k_1 and k_2 be kernels. Then the following functions are also valid kernels:

- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$,
- $k(\mathbf{x}, \mathbf{x}') = c\, k_1(\mathbf{x}, \mathbf{x}')$ for any constant $c > 0$,
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + c$ for any constant $c > 0$,
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')\, k_2(\mathbf{x}, \mathbf{x}')$,
- $k(\mathbf{x}, \mathbf{x}') = k_1(f(\mathbf{x}), f(\mathbf{x}'))$ for any mapping f.

Exercise 2.6. Prove that each of the above constructions indeed yields a valid kernel.

Finally, we remark that in practical applications, the choice of a kernel is highly problem-dependent. Designing effective kernels often requires incorporating domain-specific knowledge.

Exercise 2.7. Show that the following functions are PSD.

- $k(x, y) = \cos(x - y)$ over $\mathbb{R} \times \mathbb{R}$.
- $k(x, y) = \cos(x^2 - y^2)$ over $\mathbb{R} \times \mathbb{R}$.
- $k(x, y) = 1/(x + y)$ over $(0, +\infty) \times (0, +\infty)$. (Hint: $a^{-1} = \int_1^\infty t^{-1-a} dt$.)
- $k(x, y) = e^{-\|x-y\|_1}$ over $\mathbb{R}^d \times \mathbb{R}^d$. (Hint: use the Fourier transform.)

2.4 Representer theorem (general form)

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel with associated feature map $\phi : \mathcal{X} \to \mathcal{H}$. Consider the feature-based model

$$f(\mathbf{x}; \boldsymbol{\beta}) = \langle \boldsymbol{\beta}, \phi(\mathbf{x}) \rangle, \qquad \boldsymbol{\beta} \in \mathcal{H}.$$

We are interested in the following general regularized empirical risk minimization problem:

$$\widehat{\mathcal{R}}(\boldsymbol{\beta}) \;=\; \frac{1}{2n} \sum_{i=1}^{n} \big(f(\mathbf{x}_i; \boldsymbol{\beta}) - y_i \big)^2 + \lambda\, r(\|\boldsymbol{\beta}\|), \tag{29}$$

where $r : [0, \infty) \to [0, \infty)$ is a non-decreasing penalty function and $\lambda > 0$ is a regularization parameter, and $\|\boldsymbol{\beta}\| = \sqrt{\langle \boldsymbol{\beta}, \boldsymbol{\beta} \rangle}$ denotes the norm in $\mathcal{H}$.

Theorem 2.8 (Representer theorem)**.** *Let $\hat{\boldsymbol{\beta}}$ be a minimizer of (29). Then there exist coefficients $\hat{\mathbf{a}} \in \mathbb{R}^n$ such that*

$$f(\mathbf{x}; \hat{\boldsymbol{\beta}}) = \langle \hat{\boldsymbol{\beta}}, \phi(\mathbf{x}) \rangle = \sum_{i=1}^{n} \hat{a}_i\, k(\mathbf{x}_i, \mathbf{x}).$$

In particular, the minimizer $\hat{\boldsymbol{\beta}}$ lies in the linear span of the training features $\{\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)\}$.

Proof. Let $V_n = \operatorname{span}\{\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_n)\} \subset \mathcal{H}$. For any $\boldsymbol{\beta} \in \mathcal{H}$, decompose

$$\boldsymbol{\beta} = \boldsymbol{\beta}_{\|} + \boldsymbol{\beta}_{\perp},$$

where $\boldsymbol{\beta}_{\|} \in V_n$ and $\boldsymbol{\beta}_{\perp} \in V_n^{\perp}$. By the Pythagorean theorem, $\|\boldsymbol{\beta}\|^2 = \|\boldsymbol{\beta}_{\|}\|^2 + \|\boldsymbol{\beta}_{\perp}\|^2$. Since $r(\cdot)$ is non-decreasing, it follows that

$$r(\|\boldsymbol{\beta}\|) \;\geq\; r(\|\boldsymbol{\beta}_{\|}\|). \tag{30}$$

On the other hand, for each training point $\mathbf{x}_i$,

$$f(\mathbf{x}_i; \boldsymbol{\beta}) = \langle \boldsymbol{\beta}, \phi(\mathbf{x}_i) \rangle = \langle \boldsymbol{\beta}_{\|}, \phi(\mathbf{x}_i) \rangle + \langle \boldsymbol{\beta}_{\perp}, \phi(\mathbf{x}_i) \rangle = \langle \boldsymbol{\beta}_{\|}, \phi(\mathbf{x}_i) \rangle, \tag{31}$$

where the last equality holds because $\boldsymbol{\beta}_{\perp} \in V_n^{\perp}$.

Combining (30) and (31), we see that $\widehat{\mathcal{R}}(\boldsymbol{\beta}) \;\geq\; \widehat{\mathcal{R}}(\boldsymbol{\beta}_{\|})$. Hence, any minimizer $\hat{\boldsymbol{\beta}}$ must lie in V_n. Write $\hat{\boldsymbol{\beta}}_{\|} = \sum_{i=1}^n \hat{a}_i \phi(\mathbf{x}_i)$. Then the corresponding function can be expressed as

$$f(\mathbf{x}; \hat{\boldsymbol{\beta}}) = \langle \hat{\boldsymbol{\beta}}_{\|}, \phi(\mathbf{x}) \rangle = \sum_{i=1}^n \hat{a}_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle = \sum_{i=1}^n \hat{a}_i \, k(\mathbf{x}_i, \mathbf{x}).$$

This proves the theorem. □

This result extends Theorem 2.1 to general feature maps and regularization functions. In particular, although $\mathcal{H}$ may be infinite-dimensional, the search can be restricted to the n-dimensional span $\operatorname{span}\{k(\cdot, x_i)\}_{i=1}^n$; the solution depends only on the kernel k, not on a specific choice of feature map. This statement is known in the literature as the *representer theorem*, and it plays a fundamental role in the theory and practice of kernel methods.

Lastly, we remark that for a specific kernel, the associated feature maps are not necessarily unique. But the representer theorem shows that the solutions only depend on the kernel and is independent of the specific choice of feature maps.

2.5 Kernel ridge regression

By the representer theorem, the solution to (29) can be restricted to the form $\boldsymbol{\beta} = \sum_{j=1}^n a_j \, \phi(\mathbf{x}_j)$. Equivalently, kernel methods search for functions of the form

$$f(\mathbf{x}; \boldsymbol{\beta}) \;=\; \sum_{j=1}^n a_j \, k(\mathbf{x}_j, \mathbf{x}). \tag{32}$$

The ridge penalty then becomes

$$\|\boldsymbol{\beta}\|^2 = \Big\langle \sum_{i=1}^n a_i \phi(\mathbf{x}_i), \sum_{j=1}^n a_j \phi(\mathbf{x}_j) \Big\rangle = \sum_{i,j=1}^n a_i a_j \, k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^\top K \mathbf{a}.$$

Thus the infinite-dimensional problem (29) is equivalent to

$$\widehat{\mathcal{R}}(\mathbf{a}) = \frac{1}{2n} \sum_i \left(\sum_j a_j k(\mathbf{x}_j, \mathbf{x}_i) - y_i \right)^2 + \lambda \, r \left(\sqrt{\sum_{i,j} a_i a_j k(\mathbf{x}_i, \mathbf{x}_j)} \right)$$

$$= \frac{1}{2n}\|K\mathbf{a} - \mathbf{y}\|_2^2 + \lambda\, r\left(\sqrt{\mathbf{a}^\top K\mathbf{a}}\right). \tag{33}$$

The popular kernel ridge regression (KRR) corresponds to $r(t) = t^2/2$, where the reduced model becomes

$$\widehat{\mathcal{R}}(\mathbf{a}) = \frac{1}{2n}\|K\mathbf{a} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\mathbf{a}^\top K\mathbf{a}.$$

Similar to the standard ridge regression, KRR has a closed-form expression:

$$\hat{\mathbf{a}} = \left(\tfrac{1}{n}K + \lambda I_n\right)^{-1}\tfrac{1}{n}\mathbf{y}.$$

3 Random Feature Models

Let $(\Omega, \mathcal{F}, \pi)$ be a probability space, and let $\varphi : \mathcal{X} \times \Omega \to \mathbb{R}$ be a parametric feature function. A *random feature model* (RFM) is defined as

$$f(\mathbf{x}; \boldsymbol{\beta}) \;=\; \frac{1}{m}\sum_{j=1}^{m} \beta_j\, \varphi(\mathbf{x}; \boldsymbol{\omega}_j), \tag{34}$$

where $\{\boldsymbol{\omega}_j\}_{j=1}^m$ are i.i.d. samples from π. The name *random feature* comes from the fact that the parameters $\{\boldsymbol{\omega}_j\}$ are chosen randomly.

The ridge regression problem in the RFM setting is

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^m} \frac{1}{2n}\sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m}\beta_j\varphi(\mathbf{x}_i;\boldsymbol{\omega}_j) - y_i\right)^2 + \frac{\lambda}{2m}\|\boldsymbol{\beta}\|_2^2. \tag{35}$$

The normalization factor $1/m$ ensures that $\frac{1}{m}\|\boldsymbol{\beta}\|_2^2 = O(1)$. This formulation induces a finite-dimensional feature map

$$\hat{\phi}(\mathbf{x}) = \frac{1}{\sqrt{m}}\big(\varphi(\mathbf{x};\boldsymbol{\omega}_1), \varphi(\mathbf{x};\boldsymbol{\omega}_2), \ldots, \varphi(\mathbf{x};\boldsymbol{\omega}_m)\big)^\top \in \mathbb{R}^m.$$

Connection to kernel methods. By the representer theorem (Theorem 2.8), the RFM ridge regression problem (35) is equivalent to kernel ridge regression with the *empirical kernel*

$$\hat{k}_m(\mathbf{x}, \mathbf{x}') = \langle\hat{\phi}(\mathbf{x}), \hat{\phi}(\mathbf{x}')\rangle_{\ell^2} = \frac{1}{m}\sum_{j=1}^{m}\varphi(\mathbf{x};\boldsymbol{\omega}_j)\,\varphi(\mathbf{x}';\boldsymbol{\omega}_j). \tag{36}$$

Since $\{\boldsymbol{\omega}_j\}$ are i.i.d. from π, the law of large numbers implies that as $m \to \infty$,

$$\hat{k}_m(\mathbf{x}, \mathbf{x}') \;\to\; k(\mathbf{x}, \mathbf{x}') := \mathbb{E}_{\boldsymbol{\omega}\sim\pi}\left[\varphi(\mathbf{x};\boldsymbol{\omega})\,\varphi(\mathbf{x}';\boldsymbol{\omega})\right]. \tag{37}$$

Thus, $\hat{k}_m$ is a Monte Carlo approximation of k. Standard estimates give

$$\mathbb{E}\Big[\big\|\hat{k}_m - k\big\|^2_{L^2(\mathcal{X}\times\mathcal{X})}\Big] = \frac{\mathbf{Var}\big[\varphi(\mathbf{x};\boldsymbol{\omega})\varphi(\mathbf{x}';\boldsymbol{\omega})\big]}{m}, \tag{38}$$

so the approximation error decays at rate $O(1/\sqrt{m})$. For large m, RFMs behave similarly to kernel methods and can be interpreted as *random feature approximations* of kernel methods.

Computational motivation. RFMs were introduced by Rahimi and Recht [Rahimi and Recht, 2007] as a scalable approximation to kernel methods. For large-scale datasets with $n \gg 1$, standard kernel methods require $O(n^2)$ memory to store the kernel matrix and about $O(n^3)$ computation to invert it. While RFMs reduce these to $O(mn)$ memory and $O(m^2 n)$ computation, which is much smaller when $m \ll n$.

In summary, whenever a kernel admits the expectation form (37), one can approximate it with random features and use the finite-dimensional model (34). This leads to a natural question:

Which kernels admit such expectation representations?

We answer this next for translation-invariant kernels.

3.1 Random Fourier features

A particularly important class is the *random Fourier features* (RFFs). The basic complex-valued feature is defined as

$$\varphi(\mathbf{x}; \boldsymbol{\omega}) = e^{i \mathbf{x}^\top \boldsymbol{\omega}}.$$

In practice, one typically works in the real domain using the equivalent feature map

$$\tilde{\varphi}(\mathbf{x}; \boldsymbol{\omega}) = \begin{pmatrix} \cos(\boldsymbol{\omega}^\top \mathbf{x}) \\ \sin(\boldsymbol{\omega}^\top \mathbf{x}) \end{pmatrix}.$$

Theorem 3.1 (Bochner)**.** *A continuous function $k(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x} - \mathbf{x}')$ on $\mathbb{R}^d$ is positive semidefinite if and only if κ is the Fourier transform of a non-negative measure.*

As a consequence, if κ is the Fourier transform of a non-negative measure π, then

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x} - \mathbf{x}') &= \int_{\mathbb{R}^d} e^{i \boldsymbol{\omega}^\top (\mathbf{x} - \mathbf{x}')} \, \mathrm{d}\pi(\boldsymbol{\omega}) \\ &= \int_{\mathbb{R}^d} \varphi(\mathbf{x}; \boldsymbol{\omega}) \, \overline{\varphi(\mathbf{x}'; \boldsymbol{\omega})} \, \mathrm{d}\pi(\boldsymbol{\omega}) \\ &= \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{L^2(\pi)}, \end{aligned} \tag{39}$$

where the feature map $\phi : \mathcal{X} \to L^2(\pi)$ is given by $\phi(\mathbf{x}) = \varphi(\mathbf{x}; \cdot)$. This shows that every translation-invariant kernel of this form can be represented via random Fourier features (see [Rudin, 2017] for the full proof). Table 1 lists several popular translation-invariant kernels along with their Fourier transforms.

Remark 3.2. This provides an example where the feature space is infinite-dimensional.

Kernel	$k(\mathbf{z})$	Fourier measure $\pi(\boldsymbol{\omega})$
Gaussian	$e^{-\|\mathbf{z}\|_2^2/2}$	$\frac{1}{(2\pi)^{d/2}} e^{-\|\boldsymbol{\omega}\|_2^2/2}$
Laplace	$e^{-\|\mathbf{z}\|_1}$	$\prod_{j=1}^{d} \frac{1}{\pi(1+\omega_j^2)}$

Table 1: Examples of translation-invariant kernels and their Fourier transforms. See [Rahimi and Recht, 2007] for more details.

Example: Gaussian kernel. For the Gaussian kernel, the Fourier transform gives

$$e^{-\|\mathbf{x}-\mathbf{y}\|_2^2/2} = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} e^{-\|\boldsymbol{\omega}\|_2^2/2}\, e^{i(\mathbf{x}-\mathbf{y})\cdot\boldsymbol{\omega}}\, \mathrm{d}\boldsymbol{\omega} \tag{40}$$

$$= \mathbb{E}_{\boldsymbol{\omega}\sim\mathcal{N}(0,I_d)} \left[e^{i\boldsymbol{\omega}^\top \mathbf{x}} e^{-i\boldsymbol{\omega}^\top \mathbf{y}} \right]. \tag{41}$$

Since the kernel is real-valued, this expression reduces to

$$e^{-\|\mathbf{x}-\mathbf{y}\|_2^2/2} = \mathbb{E}_{\boldsymbol{\omega}\sim\mathcal{N}(0,I_d)} \big[\cos(\boldsymbol{\omega}^\top \mathbf{x}) \cos(\boldsymbol{\omega}^\top \mathbf{y}) + \sin(\boldsymbol{\omega}^\top \mathbf{x}) \sin(\boldsymbol{\omega}^\top \mathbf{y}) \big].$$

Thus, the Gaussian kernel can be approximated by

$$e^{-\|\mathbf{x}-\mathbf{y}\|_2^2/2} \;\approx\; \frac{1}{m} \sum_{j=1}^{m} \tilde{\varphi}(\mathbf{x};\boldsymbol{\omega}_j)^\top\, \tilde{\varphi}(\mathbf{y};\boldsymbol{\omega}_j), \qquad \boldsymbol{\omega}_j \sim \mathcal{N}(0, I_d).$$

References

[Aronszajn, 1950] Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404.

[Boyd et al., 2011] Boyd, S., Parikh, N., and Chu, E. (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc.

[Candes et al., 2006] Candes, E., Romberg, J., and Tao, T. (2006). Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509.

[Chen and Xu, 2020] Chen, L. and Xu, S. (2020). Deep neural tangent kernel and Laplace kernel have the same RKHS. In *International Conference on Learning Representations*.

[Geifman et al., 2020] Geifman, A., Yadav, A., Kasten, Y., Galun, M., Jacobs, D., and Basri, R. (2020). On the similarity between the Laplace and neural tangent kernels. *arXiv preprint arXiv:2007.01580*.

[Rahimi and Recht, 2007] Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer.

[Rudin, 2017] Rudin, W. (2017). *Fourier analysis on groups*. Courier Dover Publications.

[Steinwart and Christmann, 2008] Steinwart, I. and Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

[Vershynin, 2018] Vershynin, R. (2018). *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press.