

Lecture 5: Support Kernel Machines (SVMs) and Kernel Methods

Readings: ESL (Ch. 5.8-5.9, Ch. 12.1-12.3), ISL (Ch. 9),
Bach (Ch. 7); code

Soon Hoe Lim

September 9, 2025

Outline

- ① Support Vector Machines (SVMs)
- ② SVMs in Practice
- ③ Kernels: Mathematical Foundations
- ④ Reproducing Kernel Hilbert Spaces (RKHS)
- ⑤ The Representer Theorem
- ⑥ Kernel Methods in Practice
- ⑦ Approximations with Random Features
- ⑧ Exercises

Recap: The Support Vector Classifier (Linear)

The soft-margin support vector classifier finds an optimal linear boundary by solving:

$$\min_{\beta, \beta_0, \xi} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \xi_i$$

subject to $y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i$ and $\xi_i \geq 0$.

Proposition 1: The Dual Problem

The solution is found by solving the Wolfe dual problem for coefficients α_i :

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k \langle x_i, x_k \rangle$$

subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^N \alpha_i y_i = 0$.

The decision function is $f(x) = \text{sign}(\sum_{i=1}^N \alpha_i y_i \langle x, x_i \rangle + \beta_0)$. Notice that both the optimization and the solution depend only on **inner products** $\langle x_i, x_k \rangle$.

From Linear Classifiers to Nonlinear Boundaries

The Challenge of Non-linearity

Real-world decision boundaries are rarely linear. The SVC can be extended to handle non-linear boundaries by transforming the feature space. But this can lead to artificial separation due to overfitting.

The **Support Vector Machine (SVM)** generalizes this idea by constructing a linear boundary in a high-dimensional, transformed version of the feature space, leading to a non-linear boundary in the original space, while allowing minimal overlapping.

The Kernel Trick

The key idea is to replace the original features x with a basis expansion $h(x)$. The classifier becomes $f(x) = h(x)^T \beta + \beta_0$.

The Role of Inner Products

The dual optimization problem and the final decision function depend on the features only through inner products $\langle h(x_i), h(x_k) \rangle$.

- ▶ **Dual Objective:** $L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k \langle h(x_i), h(x_k) \rangle$.
- ▶ **Decision Function:** $f(x) = \sum_{i=1}^N \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0$.
- ▶ **Solution:** $f(x) = \sum_{i=1}^N \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0$.

Definition 1: The Kernel Trick

Instead of defining the (potentially very high-dimensional) feature map $h(x)$ explicitly, we only need to specify a **kernel function** $K(x, x')$ that computes the inner product in the feature space:

$$K(x, x') = \langle h(x), h(x') \rangle.$$

Popular Kernel Functions

► d-th Degree Polynomial:

$$K(x, x') = (1 + \langle x, x' \rangle)^d$$

This implicitly defines a feature space of polynomials of degree up to d .

► Radial Basis Function (RBF) Kernel:

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

This corresponds to an *infinite-dimensional* feature space. The parameter γ controls the width of the kernel.

► Neural Network Kernel:

$$K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$$

This kernel, with specific choices of parameters, can create decision boundaries similar to a two-layer neural network (but not guaranteed to be positive semi-definite for all κ_1, κ_2).

The Support Vector Machine (SVM)

Definition 2: The Support Vector Machine

The SVM is the non-linear classifier obtained by replacing all inner products $\langle x_i, x_k \rangle$ in the support vector classifier with a chosen kernel function $K(x_i, x_k)$.

SVM Dual Problem

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k K(x_i, x_k)$$

subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^N \alpha_i y_i = 0$.

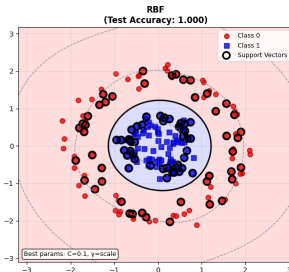
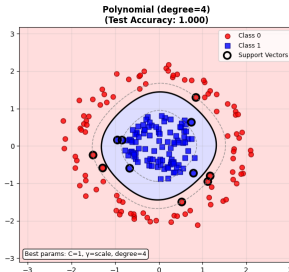
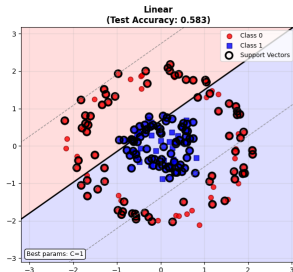
💡 The final classifier is a non-linear function of x :

$$\hat{f}(x) = \text{sign} \left(\sum_{i=1}^N \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0 \right)$$

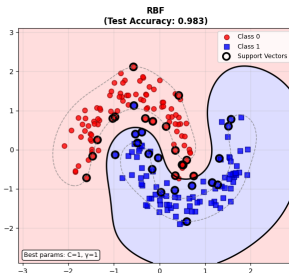
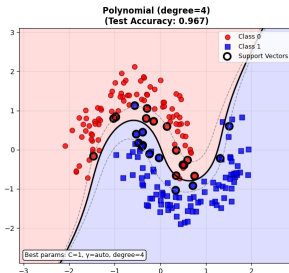
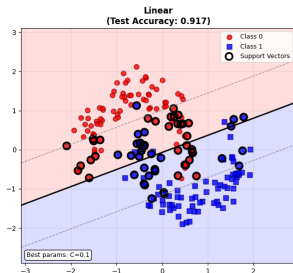
where the sum is only over the support vectors (observations for which $\hat{\alpha}_i > 0$).

Example: SVMs and the Choice of Kernels

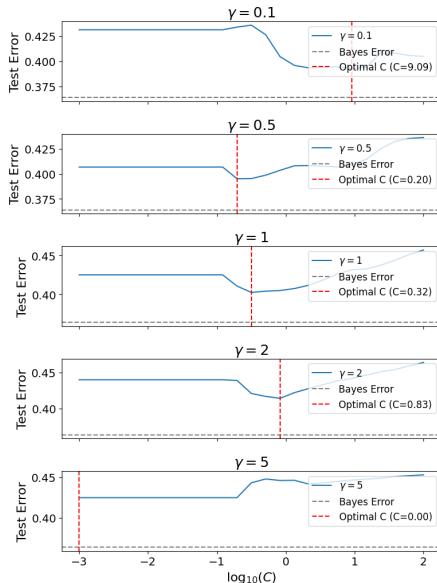
Concentric Circles: Kernel Comparison



Two Moons: Kernel Comparison



Example: SVMs with RBF Kernel of Different Widths



Not All Functions Are Valid Kernels

Given a function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, when does there exist a feature mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$ such that $K(x, x') = \langle \phi(x), \phi(x') \rangle$?

Definition 3: Positive Semi-Definite (PSD) Kernel

A function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a **positive semi-definite kernel** if:

1. **Symmetry:** $K(x, x') = K(x', x)$ for all $x, x' \in \mathcal{X}$.
2. **Positive Semi-Definiteness:** For any finite set $\{x_1, \dots, x_n\} \subset \mathcal{X}$, the Gram matrix \mathbf{K} with entries $\mathbf{K}_{ij} = K(x_i, x_j)$ satisfies:

$$\sum_{i=1}^n \sum_{j=1}^n c_i c_j K(x_i, x_j) \geq 0 \quad \text{for all } c_1, \dots, c_n \in \mathbb{R}.$$

Equivalently: All eigenvalues of every Gram matrix \mathbf{K} must be non-negative.

The Fundamental Kernel Theorem

Theorem 1: Mercer's Theorem (Finite Version)

A function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a positive semi-definite kernel if and only if there exists a Hilbert space \mathcal{H} and a mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$ such that:

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

💡 The PSD condition is both necessary and sufficient. It guarantees that the kernel trick is mathematically valid.

Kernel Verification: Examples

Verifying Polynomial Kernels

Claim: $K(x, x') = (x^T x')^d$ is PSD for any $d \geq 1$.

Proof: For any c_1, \dots, c_n and x_1, \dots, x_n :

$$\sum_{i,j} c_i c_j K(x_i, x_j) = \sum_{i,j} c_i c_j (x_i^T x_j)^d \quad (1)$$

$$= \left(\sum_i c_i x_i^T \right)^d \left(\sum_j c_j x_j \right)^d \geq 0 \quad (2)$$

since it's a $2d$ -th power of a real number.

Verifying Gaussian Kernels

Claim: $K(x, x') = \exp(-\|x - x'\|^2/(2\sigma^2))$ is PSD.

Proof idea:

- ▶ Expand: $\exp(-\|x - x'\|^2/(2\sigma^2)) = \exp(-\|x\|^2/(2\sigma^2)) \exp(x^T x'/\sigma^2) \exp(-\|x'\|^2/(2\sigma^2))$
- ▶ The middle term has Taylor series: $\exp(x^T x'/\sigma^2) = \sum_{k=0}^{\infty} \frac{(x^T x')^k}{\sigma^{2k} k!}$
- ▶ Each term $(x^T x')^k$ is PSD, positive coefficients preserve PSD property

Note: A more general proof for continuous, translation-invariant kernels relies on Bochner's Theorem, which states that such a function is PSD if and only if its Fourier transform is a non-negative measure (see Bach Ch. 7.3.3).

Operations That Preserve Kernels

Proposition 2: Kernel Closure Properties

If K_1, K_2 are PSD kernels, then the following are also PSD kernels:

1. **Positive scaling:** αK_1 for any $\alpha > 0$
2. **Sum:** $K_1 + K_2$
3. **Product:** $K_1 \cdot K_2$
4. **Pointwise limit:** $\lim_{n \rightarrow \infty} K_n$ (if it exists and is continuous)
5. **Composition with PSD function:** $f(K_1)$ where $f(t) = \sum_{n=0}^{\infty} a_n t^n$ with $a_n \geq 0$

Building Complex Kernels:

- ▶ $K(x, x') = K_1(x, x') + K_2(x, x')$ combines different types of similarity
- ▶ $K(x, x') = \exp(K_1(x, x'))$ where K_1 is PSD creates "exponential" variants
- ▶ Polynomial kernels: $(x^T x' + c)^d$

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

From Kernels to Function Spaces

The eigendecomposition construction works for finite point sets, but we want to understand kernels as defining infinite-dimensional function spaces.

Definition 4: Reproducing Kernel Hilbert Space

Given a PSD kernel K on domain \mathcal{X} , the **Reproducing Kernel Hilbert Space** \mathcal{H}_K is a Hilbert space of functions $f : \mathcal{X} \rightarrow \mathbb{R}$ 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

Theorem 2: Existence and Uniqueness of RKHS

For every positive semi-definite kernel K on \mathcal{X} , there exists a unique Reproducing Kernel Hilbert Space \mathcal{H}_K having K as its reproducing kernel.

Proposition 3: Key Properties

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

RKHS Examples

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

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

Polynomial Kernel: $K(x, x') = (1 + x^T x')^d$

\mathcal{H}_K consists of polynomials up to degree d 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
- ▶ Universal approximation: dense in $C(\mathcal{X})$ for compact \mathcal{X}
- ▶ Functions decay appropriately at infinity

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

Back to SVM: SVM as a Penalization Method

Definition 5: Hinge Loss + Penalty Formulation

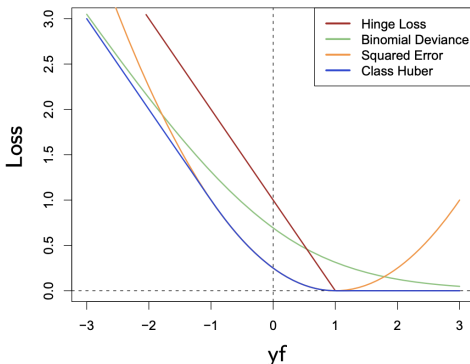
The SVM solution is equivalent to minimizing:

$$\min_{f \in \mathcal{H}_K} \left(\sum_{i=1}^N [1 - y_i f(x_i)]_+ + \lambda \|f\|_{\mathcal{H}_K}^2 \right), \quad (3)$$

- ▶ $[1 - u]_+ = \max(0, 1 - u)$ is the **hinge loss function**.
- ▶ \mathcal{H}_K is a Reproducing Kernel Hilbert Space defined by the kernel K .
- ▶ $\|f\|_{\mathcal{H}_K}^2$ is a penalty on the complexity (smoothness) of the function f .

Connection to Logistic Regression

This formulation is very similar to penalized logistic regression, which minimizes: $\sum_{i=1}^N \log(1 + e^{-y_i f(x_i)}) + \lambda \|f\|_{\mathcal{H}_K}^2$. The hinge loss and binomial deviance (logistic loss) are both convex surrogates for the 0-1 loss and produce similar classifiers.




The Key Result for Kernel Methods

Theorem 3: 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 PSD 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 $\alpha_1, \dots, \alpha_N \in \mathbb{R}$.

 **Remarkable:** Even though we're optimizing over an infinite-dimensional function space, the solution lives in an n -dimensional subspace spanned by kernel evaluations at the training points!

Computational Implications

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

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

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

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

Kernel Ridge Regression

Linear Ridge Regression: $\min_w \|\mathbf{y} - \mathbf{X}w\|^2 + \lambda \|w\|^2$

Solution: $\hat{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$

Prediction: $\hat{f}(x) = x^T \hat{w}$

The Representer Theorem allows us to substitute $f(x) = \sum_{i=1}^N \alpha_i K(x, x_i)$ into the general problem. For the squared loss, this leads to a convex optimization problem with a closed-form solution.

Definition 6: Kernel Ridge Regression

Apply the representer theorem with squared loss $L(y, f(x)) = (y - f(x))^2$:

- ▶ **Optimization problem:** $\min_{\alpha \in \mathbb{R}^N} \|\mathbf{y} - \mathbf{K}\alpha\|^2 + \lambda \alpha^T \mathbf{K} \alpha$
- ▶ **Solution:** $\hat{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$
- ▶ **Prediction:** $\hat{f}(x) = \sum_{i=1}^N \hat{\alpha}_i K(x, x_i) = \mathbf{k}(x)^T (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$, where $\mathbf{k}(x) = (K(x, x_1), \dots, K(x, x_N))^T$

Kernel Ridge Regression: Properties

Computational Complexity

- ▶ **Training:** $O(N^3)$ for matrix inversion (same as linear ridge with N features)
- ▶ **Prediction:** $O(N)$ per test point (need to evaluate N kernel functions)
- ▶ **Memory:** Store Gram matrix \mathbf{K} (N^2 elements) and coefficients $\hat{\alpha}$ (N elements)

Relationship to Linear Ridge

When can we relate the two solutions?

- ▶ If $\mathbf{K} = \mathbf{X}\mathbf{X}^T$ (i.e., $K(x_i, x_j) = x_i^T x_j$), then:

$$\hat{\alpha} = (\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I})^{-1} \mathbf{y}$$

- ▶ The linear solution is $\hat{\mathbf{w}} = \mathbf{X}^T \hat{\alpha}$
- ▶ Both give same predictions: $\hat{f}(x) = x^T \hat{\mathbf{w}} = \mathbf{k}(x)^T \hat{\alpha}$

Summary: The Power of the Kernel Viewpoint

- ▶ **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 \mathbf{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.

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 \mathbf{K} .

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

- ▶ 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.

Approximation via Random Fourier Features

Powerful way to approximate a kernel machine with a simple linear model, trading a controllable amount of accuracy for massive gain in compute efficiency.

Proposition 4: Bochner's Theorem

A continuous kernel $K(x, x')$ is positive semi-definite if and only if it is the Fourier transform of a non-negative measure. For a shift-invariant kernel $K(x, x') = K(x - x')$, this means:

$$K(x - x') = \int_{\mathbb{R}^d} p(\omega) e^{i\omega^T(x-x')} d\omega$$

where $p(\omega)$ is a non-negative probability density.

💡 We can approximate this integral with a Monte Carlo average. By sampling D frequencies ω_j from the density $p(\omega)$, we can construct an explicit, low-dimensional feature map $z(x)$ that approximates the infinite-dimensional map $h(x)$.

The Random Fourier Features Algorithm

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 $\omega_1, \dots, \omega_D$ from $\mathcal{N}(0, \frac{1}{\sigma^2}\mathbf{I})$.
- 3: **Feature Map:** Create the new feature map $z : \mathbb{R}^d \rightarrow \mathbb{R}^{2D}$: $z(x) =$

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

- 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 $K(x, x')$.

Theoretical Guarantees for Random Features

The key theoretical result shows that the performance of the linear model trained on random features converges to the performance of the full kernel machine.

Theorem 4: Approximation Guarantee

Let \hat{f}_D be the solution of Ridge Regression on D random features, and let \hat{f}_H be the solution of the full Kernel Ridge Regression. Then,

$$\mathbb{E}[R(\hat{f}_D)] - \mathbb{E}[R(\hat{f}_H)] = O\left(\frac{1}{\sqrt{D}}\right).$$

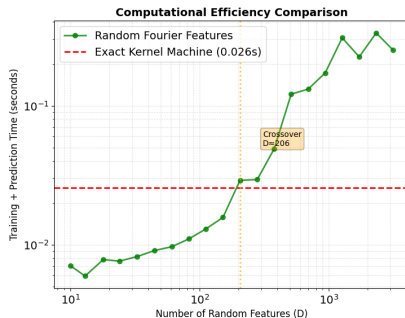
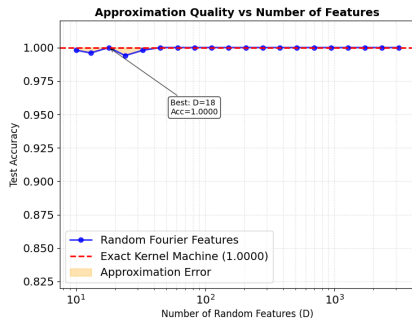
💡 A powerful result: by increasing D , we can get arbitrarily close to the performance of the exact kernel method. The computational cost is now only $O(ND^2 + D^3)$, which is much better than $O(N^3)$ if $D \ll N$.

Example: Approximating a Kernel Machine

Experiment Setup. 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 won the 2017 NIPS Test of Time Award:

<https://eecs.berkeley.edu/news/ben-recht-wins-nips-test-time-award/>

Exercise 1

1. Solve Exercise 12.10 in ESL.
2. Show that if $k : \mathcal{X} \rightarrow \mathcal{X}$ is a positive-definite kernel, then so is the function $(x, x') \mapsto e^{k(x, x')}$.
3. (Mercer kernels) Solve Exercise 7.6 in Bach.
4. (Random feature expansion of Mercer kernels) Solve Exercise 7.10 in Bach.
5. Revisit Exercise 2 in Lecture 3. Implement SVMs using different kernel functions and compare the results to the earlier methods.