

Kernel Methods

Aayush Bajaj

July 10, 2026

Contents

1	feature maps and the kernel trick	1
2	which functions are kernels? mercer and psd	2
3	the common kernels and their feature spaces	2
4	the rkhs, in one careful paragraph	3
5	the representer theorem	3
6	kernel ridge regression, derived	4
7	kernel ridge from scratch	5
8	the wider family	6
9	see also	6

kernel methods are the great arbitrage of classical machine learning: keep the algorithm linear — with all its convexity and closed forms — but run it in a feature space so large it can bend around anything, and never pay for that space explicitly. one identity powers everything: if your algorithm touches the data only through inner products, you may replace every $\langle x, x' \rangle$ with a kernel $k(x, x')$ and thereby work in the implicit feature space of k — possibly infinite-dimensional — at the cost of an $n \times n$ matrix.

the whole business is sometimes summarised as: linear models did nothing wrong, your coordinates did

1 feature maps and the kernel trick

a linear model on raw inputs $x \in \mathbb{R}^p$ can only draw hyperplanes. the classical fix is a **feature map** $\varphi : \mathcal{X} \rightarrow \mathcal{H}$ into a higher-dimensional space, and a linear model *there*: $f(x) = \langle w, \varphi(x) \rangle$. the problem is cost — mapping to $\binom{p+d}{d}$ monomial features explicitly is punishing housekeeping — polynomial in p for fixed degree, but growing like p^d , exponential in the degree itself.

the **kernel trick** removes it. define

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}, \quad (1)$$

and notice that many algorithms (ridge regression, svms, pca, perceptron, k-means) can be rewritten so the data appear *only* inside pairwise inner products. then φ never needs to be computed, stored, or even finite-dimensional — you only ever evaluate

k . the object you do pay for is the **gram matrix** $K \in \mathbb{R}^{n \times n}$, $K_{ij} = k(x_i, x_j)$: kernel methods trade dependence on the feature dimension for dependence on the sample size.

hence their decline in the billion-sample era: $O(n^2)$ memory and $O(n^3)$ solves age badly, which is what random features and inducing-point approximations patch

a two-line example makes the trick concrete. for $x, z \in \mathbb{R}^2$,

$$(x^\top z)^2 = (x_1 z_1 + x_2 z_2)^2 = \langle (x_1^2, \sqrt{2} x_1 x_2, x_2^2), (z_1^2, \sqrt{2} z_1 z_2, z_2^2) \rangle, \quad (2)$$

so the kernel $k(x, z) = (x^\top z)^2$ is an inner product in the 3-dimensional space of degree-2 monomials — evaluated in $O(p)$ work rather than by touching all $O(p^2)$ coordinates, a gap that becomes $O(p)$ versus $\binom{p+d}{d}$ for general degree d .

2 which functions are kernels? mercer and psd

not every symmetric function of two arguments is an inner product in disguise. the characterisation:

- k is **positive semidefinite** (psd) if for every finite set $\{x_1, \dots, x_n\}$ and every $c \in \mathbb{R}^n$, $\sum_{i,j} c_i c_j k(x_i, x_j) \geq 0$ — i.e. every gram matrix it generates is psd.
- (**moore–aronszajn**) k is psd if and only if it is the inner-product kernel of *some* feature map into *some* hilbert space. psd-ness is the exact price of admission.
- (**merc**) for continuous symmetric psd k on a compact domain, the associated integral operator has eigenfunctions e_i and non-negative eigenvalues λ_i with

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i e_i(x) e_i(x'), \quad (3)$$

converging uniformly — an explicit (countable) feature map $\varphi(x) = (\sqrt{\lambda_i} e_i(x))_i$, the functional-analysis analogue of diagonalising a psd matrix (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

psd-ness survives the useful algebra, which is how kernels are engineered in practice: if k_1, k_2 are psd then so are $k_1 + k_2$, αk_1 for $\alpha \geq 0$, $k_1 \cdot k_2$, $f(x)f(x')k_1(x, x')$ for any f , and pointwise limits. sums say “or-like similarity”, products say “and-like”; this closure is the origin of kernels on strings, graphs, and sets — the input space never needed to be a vector space at all.

3 the common kernels and their feature spaces

kernel	$k(x, x')$	implicit feature space
linear	$x^\top x'$	the inputs themselves ($\varphi = \text{id}$)
polynomial	$(x^\top x' + c)^d$	all monomials of degree $\leq d$ when $c > 0$ (with $c = 0$, only degree exactly d): dimension $\binom{p+d}{d}$
rbf / gaussian	$\exp(-\ x - x'\ ^2 / 2\ell^2)$	infinite-dimensional

the rbf claim deserves its one line of proof. for $p = 1$ and $2\ell^2 = 1$,

$$e^{-(x-z)^2} = e^{-x^2} e^{-z^2} e^{2xz} = e^{-x^2} e^{-z^2} \sum_{j=0}^{\infty} \frac{(2xz)^j}{j!} = \sum_{j=0}^{\infty} \underbrace{\sqrt{\frac{2^j}{j!}} x^j e^{-x^2}}_{\varphi_j(x)} \cdot \varphi_j(z), \quad (4)$$

an inner product of infinitely many weighted-monomial features: the rbf kernel is a polynomial kernel of *every* degree at once, with factorially decaying weights.

consequences: the gram matrix of distinct points is strictly positive definite, rbf machines can interpolate any labelling (infinite vc-style capacity — regularisation is not optional), and the lengthscale ℓ is the bias–variance dial: small ℓ approaches a lookup table, large ℓ approaches a constant. in high ambient dimension the pairwise distances that feed the rbf concentrate, flattening K toward uninformative — the failure mode detailed in [curse of dimensionality](#).

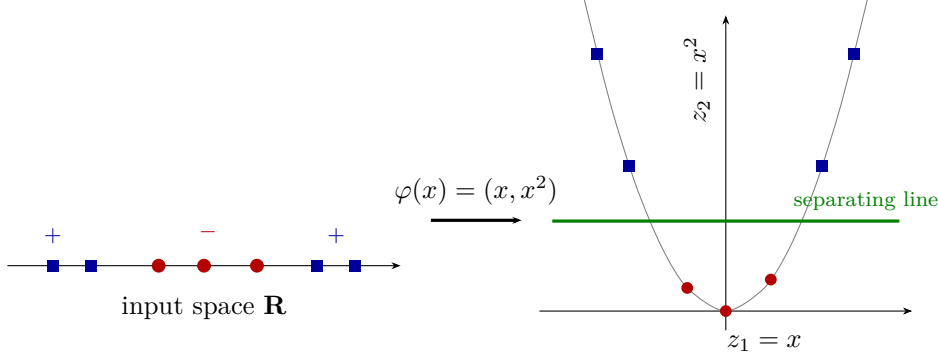


Figure 1: the lift: 1d data with negatives (circles) flanked by positives (squares) — no threshold on the line separates them. the map $\varphi(x) = (x, x^2)$ places every point on a parabola, where one straight line does the job.

4 the rkhs, in one careful paragraph

a **reproducing kernel hilbert space** $\mathcal{H}_k$ is a hilbert space whose elements are *functions* $f : \mathcal{X} \rightarrow \mathbb{R}$, with the defining property that evaluation at a point is a continuous linear functional — knowing that f and g are close in the norm of $\mathcal{H}_k$ guarantees $f(x)$ and $g(x)$ are close at every single x (false in L^2 , where functions have no pointwise values at all). by the riesz representation theorem each evaluation functional is an inner product with some element, and that element is the kernel section: $f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}_k}$, the **reproducing property**, from which $k(x, x') = \langle k(\cdot, x), k(\cdot, x') \rangle$ — so the canonical feature map is $\varphi(x) = k(\cdot, x)$, embedding each point as a function. moore–aronszajn says this construction succeeds for every psd k and yields a *unique* $\mathcal{H}_k$: the completion of finite sums $\sum_i \alpha_i k(\cdot, x_i)$ under the induced inner product. the norm $\|f\|_{\mathcal{H}_k}$ is a smoothness functional measured against the kernel — for the rbf kernel it heavily taxes high-frequency components — which is precisely what makes it the natural regulariser below (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

5 the representer theorem

theorem (kimeldorf–wahba; schölkopf–herbrich–smola). let k be a psd kernel with rkhs $\mathcal{H}_k$, and consider

$$\min_{f \in \mathcal{H}_k} L(y_1, f(x_1), \dots, y_n, f(x_n)) + \Omega(\|f\|_{\mathcal{H}_k}), \quad (5)$$

for an arbitrary loss L that touches f only through its values at the n training points and a strictly increasing Ω . then every minimiser has the form

$$f^*(\cdot) = \sum_{i=1}^n \alpha_i k(\cdot, x_i). \quad (6)$$

proof sketch. decompose $f = f_{\parallel} + f_{\perp}$, the orthogonal projection onto $V = \text{span}\{k(\cdot, x_i)\}_{i=1}^n$ plus its complement. by the reproducing property, $f(x_j) = \langle f, k(\cdot, x_j) \rangle = \langle f_{\parallel}, k(\cdot, x_j) \rangle$ — the perpendicular part is *invisible* to every training evaluation, so the loss term is unchanged by deleting it. but pythagoras gives $\|f\|^2 = \|f_{\parallel}\|^2 + \|f_{\perp}\|^2$, so Ω strictly prefers $f_{\perp} = 0$ unless $f_{\perp}$ was already zero. hence any minimiser lies in V . ■

the punchline: an optimisation over an infinite-dimensional function space collapses, exactly, to n coefficients. this is why svms, kernel ridge, and smoothing splines are finite computations at all — and the sparsity of hinge loss (see [loss functions](#)) then zeroes most of the α_i , leaving the support vectors (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020).

6 kernel ridge regression, derived

apply the representer theorem to squared loss with $\Omega = \lambda \|f\|_{\mathcal{H}_k}^2$. substituting $f = \sum_j \alpha_j k(\cdot, x_j)$, and using the reproducing property twice — $f(x_i) = (K\alpha)_i$ and $\|f\|^2 = \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) = \alpha^\top K \alpha$:

$$J(\alpha) = \|y - K\alpha\|^2 + \lambda \alpha^\top K \alpha. \quad (7)$$

set the gradient to zero:

$$\begin{aligned} \nabla_{\alpha} J &= -2K(y - K\alpha) + 2\lambda K\alpha = 0 \\ \implies K((K + \lambda I)\alpha - y) &= 0 \iff \alpha^* = (K + \lambda I)^{-1}y, \end{aligned}$$

and $K + \lambda I$ is invertible for $\lambda > 0$ since $K \succeq 0$. the fitted function and its prediction at a new point:

$$\hat{f}(x_*) = \sum_{i=1}^n \alpha_i^* k(x_*, x_i) = k_*^\top (K + \lambda I)^{-1} y, \quad k_* = (k(x_*, x_i))_{i=1}^n. \quad (8)$$

remarks worth keeping:

- with the linear kernel this reproduces ordinary ridge regression via the identity $(X^\top X + \lambda I)^{-1} X^\top = X^\top (X X^\top + \lambda I)^{-1}$ — the primal solves a $p \times p$ system, the dual an $n \times n$ one; kernelisation is just choosing the dual and then generalising the inner product.
- $\hat{f}$ is a **linear smoother**: predictions are $S_\lambda y$ with $S_\lambda = K(K + \lambda I)^{-1}$, so effective degrees of freedom $\text{tr}(S_\lambda) = \sum_i d_i / (d_i + \lambda)$ in terms of the eigenvalues d_i of K — λ continuously dials model complexity, and the [bias-variance decomposition](#) applies verbatim (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).
- everything here is $O(n^3)$ once; prediction is $O(n)$ per point.¹

¹the smoother matrix also yields the shortcut $\text{loocv} = \frac{1}{n} \sum_i ((y_i - \hat{y}_i) / (1 - S_{ii}))^2$, so leave-one-out cross validation for kernel ridge costs one fit, not n .

7 kernel ridge from scratch

fit $f(x) = \sin 2\pi x$ from 40 noisy points with an rbf kernel, sweeping the two hyper-parameters. noise sd is 0.25, so no method should beat test rmse ≈ 0.25 .

```
1 import numpy as np
2
3 rng = np.random.default_rng(3)
4
5 f = lambda x: np.sin(2 * np.pi * x)
6 n = 40
7 x_tr = rng.uniform(0, 1, n); y_tr = f(x_tr) + rng.normal(0, 0.25, n)
8 x_te = rng.uniform(0, 1, 200); y_te = f(x_te) + rng.normal(0, 0.25, 200)
9
10 def rbf(a, b, ell):
11     return np.exp(-(a[:, None] - b[None, :])**2 / (2 * ell**2))
12
13 def krr_fit_predict(ell, lam):
14     K = rbf(x_tr, x_tr, ell) # n x n gram matrix
15     alpha = np.linalg.solve(K + lam * np.eye(n), y_tr)
16     return rbf(x_te, x_tr, ell) @ alpha # k(x*, x_i) alpha
17
18 rmse = lambda a, b: np.sqrt(np.mean((a - b)**2))
19
20 # baseline: ridge on raw x (a line can't do a sine)
21 X = np.stack([np.ones(n), x_tr], axis=1)
22 w = np.linalg.solve(X.T @ X + 1e-3 * np.eye(2), X.T @ y_tr)
23 lin_pred = np.stack([np.ones(200), x_te], axis=1) @ w
24 print(f"linear ridge test rmse = {rmse(lin_pred, y_te):.4f}")
25
26 print(f"{'ell':>6} {'lambda':>8} {'test rmse':>10}")
27 for ell in (0.02, 0.1, 0.5):
28     for lam in (1e-6, 1e-2, 1.0):
29         print(f"{'ell':>6} {'lam':>8} {'rmse(krr_fit_predict(ell, lam),",
30               ↪ y_te):>10.4f}")
31 print(f"\nbayes floor (noise sd) = 0.25")
```

linear ridge test rmse = 0.4917

ell	lambda	test rmse
0.02	1e-06	9.1673
0.02	0.01	0.3858
0.02	1.0	0.4158
0.1	1e-06	0.3662
0.1	0.01	0.3020
0.1	1.0	0.2930
0.5	1e-06	0.3040
0.5	0.01	0.2683
0.5	1.0	0.4216

bayes floor (noise sd) = 0.25

the table is the bias-variance surface in miniature: tiny lengthscale with near-zero regularisation interpolates the noise and detonates (rmse 9.17 — the gram matrix is nearly singular and α explodes); the tuned corner $(\ell, \lambda) = (0.5, 10^{-2})$ sits at 0.268, within shouting distance of the 0.25 floor; over-regularising at the same lengthscale ($\lambda = 1$) shrinks the fit toward zero and bias takes over again. the linear baseline, unable to represent a sine at all, is pure bias at 0.49. hyperparameter selection here is a job for [cross validation](#).

8 the wider family

the same dual rewrite kernelises most of the classical toolbox:

- **svm**: hinge loss + rkhs norm; the representer expansion becomes sparse in the support vectors. the standard treatment is ch. 12 of (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009) and ch. 12 of (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020).
- **kernel pca**: eigendecompose the (centred) gram matrix instead of the covariance; nonlinear components for free.
- **smoothing splines**: penalised regression in the rkhs of a spline kernel — the representer theorem is why the optimiser over *all* twice-differentiable functions is a finite spline with knots at the data (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).
- **gaussian processes**: read k as a covariance function instead of an inner product; the gp posterior mean under gaussian noise σ^2 is exactly kernel ridge with $\lambda = \sigma^2$.²

choosing the kernel is choosing the inductive bias — the similarity structure you assert before seeing data. **the no free lunch theorem** guarantees there is no universally correct choice; the kernel is where you spend your assumption budget, in public.

9 see also

- **loss functions** — hinge + kernel = svm; squared + kernel = the derivation above
- **the bias-variance decomposition** — ℓ and λ are both dials on the same u-curve
- **curse of dimensionality** — what happens to rbf similarity in high dimension
- **cross validation** — how (ℓ, λ) actually get chosen
- **the no free lunch theorem** — the kernel as declared inductive bias

²the gp gives the same mean plus a posterior variance $k(x_*, x_*) - k_*^\top (K + \sigma^2 I)^{-1} k_*$ — uncertainty for free, which is the practical argument for the bayesian reading. see ch. 6 of (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020) for the gaussian machinery.

references

Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon (2020). *Mathematics for Machine Learning*, Cambridge University Press. Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome (2009). *The Elements of Statistical Learning*, Springer.