Kernel Methods

Nonlinear feature maps, valid kernels, and kernel SVM

Kangning Cui
City University of Hong Kong (Dongguan)

Semester A, 2026/27

Final examination

- **Date:** Monday, 14 December 2026.
- **Time:** 09:00–12:00 (3 hours).
- **Format:** Closed-book.
- **Paper:** Five written questions, 20 marks each (100 marks total).

Group project

- **Presentations:** Begin in two weeks, on 30 November.
- **Slides:** Each group uploads one slide deck to Canvas by 27 November 2026.
- **Groups:** During today's first break, come to my desk to record your group number and members on my sheet.

Learning objectives

- Explain how kernels handle nonlinear patterns.
- Calculate kernel values and use them in familiar algorithms.
- Understand kernel choices and their practical costs.

Linear SVM recap

The maximum-margin rule from Week 10 is useful, but its boundary is still a hyperplane.

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b.$$

A hyperplane cannot represent every class boundary. Curved patterns need a more flexible geometry. We will keep the SVM principle and change the feature representation.

Nonlinear features

CS5487 ▪ Semester A, 2026/27

Linear separation

To see the limitation of linear SVM, consider the boundary for two input features:

$$w_1 x_1 + w_2 x_2 + b = 0.$$

It is a straight line. A surrounding ring or an alternating class pattern cannot always be separated this way. A nonlinear feature transformation can change the boundary shape.

XOR geometry

An XOR dataset is a small example in which a straight-line boundary is insufficient.

Class $+1$	Class -1
$(1,1),\ (-1,-1)$	$(1,-1),\ (-1,1)$

Opposite corners have the same label. No straight line separates both positive points from both negative points.

XOR transformation

For these four points, the sign of the coordinate product is exactly what distinguishes the classes.

$$z = x_1 x_2.$$

For the four XOR points:

(x_1, x_2)	$(1,1)$	$(-1,-1)$	$(1,-1)$	$(-1,1)$
z	1	1	-1	-1

The simple rule $\hat{y} = \text{sign}(z)$ now separates the classes. A nonlinear transformation made a linear threshold sufficient.

Feature maps

The product used for XOR is one example of a general feature transformation. For an input $\boldsymbol{x} \in \mathbb{R}^d$, write

$$\boldsymbol{x} \longmapsto \phi(\boldsymbol{x}).$$

A linear model in that new space is

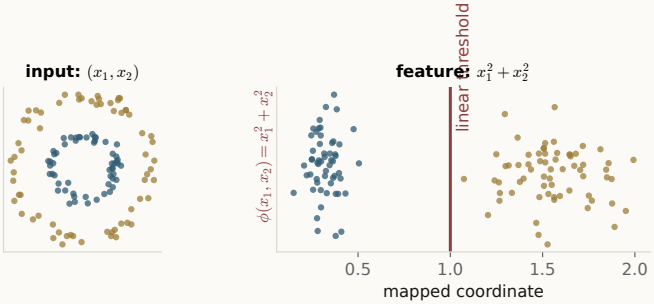
$$\boxed{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathsf{T}} \phi(\boldsymbol{x}) + b.}$$

The score is linear in $\phi(\boldsymbol{x})$, but may be nonlinear in the original input $\boldsymbol{x}$.

Circular separation

Other patterns need different features. In the figure, inner points have smaller squared radius than points on the outer ring.

$$\phi(\boldsymbol{x}) = x_1^2 + x_2^2, \qquad \boxed{\phi(\boldsymbol{x}) = c \iff x_1^2 + x_2^2 = c.}$$



Circular threshold

The circular boundary $x_1^2 + x_2^2 = 4$ becomes a threshold on squared radius.

Input $\boldsymbol{x}$	$x_1^2 + x_2^2$	Location
$(1, 1)$	2	Inside
$(3, 0)$	9	Outside
$(0, -2)$	4	On boundary

Only one transformed feature is needed to apply the circular rule.

Polynomial features

Designing features one at a time is inconvenient when several interactions may matter.

$$\phi(\boldsymbol{x}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 \end{bmatrix}^{\mathsf{T}}.$$

It expands two coordinates into six.

- $x_1 x_2$ captures an interaction.
- x_1^2 and x_2^2 capture curvature.

More degrees provide more flexibility, but also more features.

Feature growth

A full polynomial expansion grows quickly with the number d of input features and maximum degree q:

$$\binom{d+q}{q} \quad \text{features.}$$

With $d = 100$ and $q = 2$:

$$\binom{102}{2} = 5151.$$

Constructing high-dimensional feature vectors can be costly. The next step is to compute inner products without storing every feature.

Inner-product shortcut

To avoid storing many nonlinear coordinates, look at what linear algorithms actually calculate.

$$\phi(\boldsymbol{x})^{\mathsf{T}}\phi(\boldsymbol{z}).$$

If this quantity can be computed directly from the original inputs, explicit coordinates may be unnecessary.

$$\boxed{k(\boldsymbol{x}, \boldsymbol{z}) = \phi(\boldsymbol{x})^{\mathsf{T}}\phi(\boldsymbol{z}).}$$

This equality is the basis of the kernel trick.

Quadratic identity

A short algebraic identity demonstrates how to skip an explicit nonlinear feature vector.

$$\phi(x) = \begin{bmatrix} 1 \\ \sqrt{2}x \\ x^2 \end{bmatrix}.$$

Then

$$\begin{aligned} \phi(x)^{\mathsf{T}}\phi(z) &= 1 + 2xz + x^2 z^2 \\ &= \boxed{(1 + xz)^2}. \end{aligned}$$

The shorter formula is an exact feature-space inner product.

Numerical verification

For $x = 1$ and $z = 2$, the explicit and kernel calculations should agree.

$$k(1, 2) = (1 + 2)^2 = \boxed{9}.$$

From the explicit features,

$$\phi(1) = \begin{bmatrix} 1 \\ \sqrt{2} \\ 1 \end{bmatrix}, \quad \phi(2) = \begin{bmatrix} 1 \\ 2\sqrt{2} \\ 4 \end{bmatrix},$$

so $\phi(1)^{\mathsf{T}}\phi(2) = 1 + 4 + 4 = 9$.

Multivariate polynomial map

For two-dimensional vectors $\boldsymbol{x} = (x_1, x_2)^\mathsf{T}$ and $\boldsymbol{z}$, the shortcut also contains interactions.

$$k(\boldsymbol{x}, \boldsymbol{z}) = (1 + \boldsymbol{x}^\mathsf{T}\boldsymbol{z})^2.$$

One feature map is

$$\phi(\boldsymbol{x}) = \begin{bmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^\mathsf{T}.$$

The kernel computes its six-dimensional inner product from only the two original coordinates.

Kernel computation

The benefit of the kernel trick is a cheaper route to the same inner product.

$$\begin{aligned}
&\text{Explicit:} \quad \boldsymbol{x}, \boldsymbol{z} \;\to\; \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \;\to\; \phi(\boldsymbol{x})^{\mathsf{T}} \phi(\boldsymbol{z}), \\
&\text{Implicit:} \quad \boldsymbol{x}, \boldsymbol{z} \;\to\; \boxed{k(\boldsymbol{x}, \boldsymbol{z})}.
\end{aligned}$$

The implicit route is useful when the feature map is large, provided the algorithm depends on inner products.

Kernel basics

CS5487 ▪ Semester A, 2026/27

Kernel representation

We now formalize the inner-product function used in the preceding example.

$$\boxed{k(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle}$$

for some feature map ϕ. The feature space may be finite-dimensional, very large, or infinite-dimensional. We need not construct it explicitly.

Kernel similarity

A kernel often measures similarity, but its value can also depend on the lengths of the feature vectors. For example,

$$k(\boldsymbol{x}, \boldsymbol{x}) = \|\phi(\boldsymbol{x})\|_2^2.$$

Before choosing a kernel, we must understand the matrix formed by its pairwise values.

Gram matrices

A learning algorithm compares n training observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in pairs.

$$K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

The **Gram matrix** is

$$\boxed{\boldsymbol{K} = [K_{ij}]_{i,j=1}^{n}.}$$

It contains every pairwise inner product in the implicit feature space.

Two-point Gram matrix

Take $x_1 = 1$, $x_2 = 2$, and $k(x, z) = (1 + xz)^2$ to build a two-point Gram matrix.

$$\begin{aligned} K_{11} &= (1+1)^2 = 4, \\ K_{12} = K_{21} &= (1+2)^2 = 9, \\ K_{22} &= (1+4)^2 = 25. \end{aligned}$$

Therefore,

$$\boxed{\boldsymbol{K} = \begin{bmatrix} 4 & 9 \\ 9 & 25 \end{bmatrix}.}$$

Three-point Gram matrix

For $x_1 = 0$, $x_2 = 1$, $x_3 = 2$, use the same kernel to build the three-point Gram matrix.

$$\boxed{\boldsymbol{K} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 9 \\ 1 & 9 & 25 \end{bmatrix}.}$$

For instance, $K_{23} = (1 + 1 \cdot 2)^2 = 9$. Each row compares one point with all three observations.

Kernel validity

A kernel must represent inner products in some feature space. For any finite dataset, its Gram matrix $\boldsymbol{K}$ must be symmetric and **positive semidefinite (PSD)**:

$$\boxed{\boldsymbol{c}^{\mathsf{T}}\boldsymbol{K}\boldsymbol{c} \geq 0 \qquad \text{for all } \boldsymbol{c} \in \mathbb{R}^n.}$$

This condition keeps kernel values consistent with an inner-product geometry.

A validity check

Symmetry alone cannot guarantee a valid kernel matrix. Consider

$$\boldsymbol{S} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \qquad \boldsymbol{c} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Then

$$\boxed{\boldsymbol{c}^{\mathsf{T}} \boldsymbol{S} \boldsymbol{c} = -2 < 0.}$$

This matrix cannot be a valid kernel Gram matrix. One negative result is enough to reject it.

Kernel families

Once validity is understood, we can choose kernels to encode different kinds of structure.

Kernel	Formula
Linear	$\boldsymbol{x}^\mathsf{T}\boldsymbol{z}$
Polynomial	$(c + \boldsymbol{x}^\mathsf{T}\boldsymbol{z})^q$
RBF	$\exp(-\gamma\|\boldsymbol{x} - \boldsymbol{z}\|_2^2)$
Laplacian	$\exp(-\|\boldsymbol{x} - \boldsymbol{z}\|_1/\sigma)$

Use $c \geq 0$, integer $q \geq 1$, $\gamma > 0$, and $\sigma > 0$ for the corresponding standard families.

Linear kernel

The linear kernel is the simplest reference point for nonlinear alternatives:

$$\boxed{k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}^\mathsf{T} \boldsymbol{z}.}$$

Its feature map is $\phi(\boldsymbol{x}) = \boldsymbol{x}$, so it preserves the original geometry. A nonlinear kernel is useful only when this geometry is insufficient.

Polynomial kernel

When feature interactions are important, polynomial kernels provide them implicitly.

$$\boxed{k(\boldsymbol{x}, \boldsymbol{z}) = (c + \boldsymbol{x}^\mathsf{T}\boldsymbol{z})^q.}$$

A degree-two kernel includes quadratic interactions; larger degrees allow higher-order interactions. The polynomial kernel is useful when products of features carry information that separate original coordinates miss.

Polynomial example

Take $\boldsymbol{x} = (1, 2)^\mathsf{T}$, $\boldsymbol{z} = (2, -1)^\mathsf{T}$, $c = 1$, and $q = 2$. Kernel evaluation uses the original dot product:

$$\boldsymbol{x}^\mathsf{T}\boldsymbol{z} = 1 \cdot 2 + 2(-1) = 0.$$

Therefore,

$$\boxed{k(\boldsymbol{x}, \boldsymbol{z}) = (1 + 0)^2 = 1.}$$

The calculation uses only the original dot product; no quadratic feature vector is needed.

RBF kernel

Polynomial kernels use products; for local patterns, similarity can instead depend on distance.

$$\boxed{k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{z}\|_2^2\right), \quad \gamma > 0.}$$

Identical inputs have kernel value 1. Similarity decreases as Euclidean distance increases. The parameter γ determines how quickly that decrease occurs.

RBF Gram example

The Gram matrix changes when we choose another kernel. Take $x_1 = 0$, $x_2 = 1$, $x_3 = 2$ and $k(x, z) = e^{-\gamma(x-z)^2}$ with $\gamma = \ln 2$.

$$\boldsymbol{K} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{16} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{16} & \frac{1}{2} & 1 \end{bmatrix}.$$

Adjacent points have similarity $1/2$; points two units apart have similarity $1/16$.

RBF similarity

With $\gamma = 1$ and $d = \|\boldsymbol{x} - \boldsymbol{z}\|_2$, a few distances show how quickly similarity decreases.

Distance d	d^2	$k = e^{-d^2}$
0	0	1
1	1	e^{-1}
2	4	e^{-4}

Similarity drops much faster than distance grows.

RBF bandwidth

The scale parameter determines how far one observation influences another.

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

The two parameterizations satisfy

$$\boxed{\gamma = \frac{1}{2\sigma^2}.}$$

Large γ means narrow similarity; small γ means broad similarity.

Different kernel values

The same two inputs can look different under different kernels. Take $x = 1$ and $z = 2$.

Kernel	Definition	$k(1, 2)$
Linear	xz	2
Polynomial	$(1 + xz)^2$	9
RBF	$e^{-(\ln 2)(x-z)^2}$	$1/2$

Raw kernel values have different scales; compare similarities within a chosen kernel.

Feature-space distance

Inner products also determine distances, so a kernel changes the distance geometry.

$$\begin{aligned} d_\phi^2(\boldsymbol{x}, \boldsymbol{z}) &= \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{z})\|_2^2 \\ &= \boxed{k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{z}, \boldsymbol{z}) - 2k(\boldsymbol{x}, \boldsymbol{z})}. \end{aligned}$$

We can measure this distance without computing $\phi(\boldsymbol{x})$.

Distance example

For $x = 1$, $z = 2$, and $k(x, z) = (1 + xz)^2$, compare original and feature-space distances.

$$k(1,1) = 4, \quad k(2,2) = 25, \quad k(1,2) = 9.$$

Hence

$$\boxed{d_\phi^2(1,2) = 4 + 25 - 2(9) = 11.}$$

The original squared distance is only $(1-2)^2 = 1$. A kernel changes the geometry measured by an algorithm.

RBF feature distance

For RBF kernels, the feature-space distance takes a particularly simple form.

$$\boxed{d_\phi^2(\boldsymbol{x}, \boldsymbol{z}) = 2 - 2k(\boldsymbol{x}, \boldsymbol{z}).}$$

- Identical inputs: $d_\phi^2 = 0$.
- Very distant inputs: $d_\phi^2 \to 2$.

RBF feature-space distances are bounded, even when Euclidean input distance is not.

Kernel combinations

If k_1, k_2 are valid kernels, combining them can incorporate two kinds of similarity. For $a, b \geq 0$:

$$\boxed{k = ak_1 + bk_2}$$

is also valid. Their pointwise product $k_1 k_2$ is valid as well. For example, $0.7k_{\text{text}} + 0.3k_{\text{image}}$ combines two sources of similarity.

Kernel SVM

CS5487 ▪ Semester A, 2026/27

Linear SVM dual

The SVM dual from Week 10 is the natural entry point because it uses only dot products.

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j,$$

subject to

$$0 \le \alpha_i \le C, \qquad \sum_i \alpha_i y_i = 0.$$

Training observations enter the objective only through their dot products.

Feature-space SVM

Applying SVM to nonlinear features changes only the dot products in the dual. In its dual objective, each dot product becomes

$$\phi(\boldsymbol{x}_i)^\mathsf{T}\phi(\boldsymbol{x}_j).$$

The kernel replaces this calculation exactly:

$$\boxed{\phi(\boldsymbol{x}_i)^\mathsf{T}\phi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j).}$$

We retain the SVM constraints and maximum-margin principle.

Kernel SVM dual

Replacing these dot products by kernel evaluations gives the kernel SVM optimization.

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

with constraints

$$0 \leq \alpha_i \leq C, \qquad \sum_i \alpha_i y_i = 0.$$

The dual variables α_i correspond to training observations; the kernel selects their feature-space geometry.

Kernel SVM prediction

After finding the dual coefficients, the classifier must also evaluate a new observation.

$$f(\boldsymbol{x}) = \sum_i \alpha_i y_i \phi(\boldsymbol{x}_i)^\mathsf{T} \phi(\boldsymbol{x}) + b.$$

Therefore

$$\boxed{f(\boldsymbol{x}) = \sum_i \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b.}$$

The predicted class is $\operatorname{sign}(f(\boldsymbol{x}))$ when $f(\boldsymbol{x}) \neq 0$.

Support vectors

The dual representation has a practical advantage: coefficients equal to zero can be omitted.

$$f(\boldsymbol{x}) = \sum_{i:\alpha_i > 0} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b.$$

The observations with $\alpha_i > 0$ are **support vectors**. The number of support vectors affects the cost of predicting a new observation.

SVM intercept

The dual supplies α_i, but prediction also needs b. For a margin support vector with $0 < \alpha_i < C$, KKT gives

$$y_i f(\boldsymbol{x}_i) = 1.$$

Since $y_i \in \{-1, +1\}$,

$$\boxed{b = y_i - \sum_j \alpha_j y_j k(\boldsymbol{x}_j, \boldsymbol{x}_i).}$$

The kernel prediction formula therefore also provides the SVM intercept.

Prediction example

Consider a score with two support vectors. Their labels and multipliers are:

$$(\alpha_1, y_1) = (\tfrac{1}{2}, -1), \qquad (\alpha_2, y_2) = (\tfrac{1}{2}, +1), \qquad b = 0.$$

For a new input $\boldsymbol{x}$, suppose

$$k(\boldsymbol{x}_1, \boldsymbol{x}) = \tfrac{1}{4}, \qquad k(\boldsymbol{x}_2, \boldsymbol{x}) = \tfrac{3}{4}.$$

Then

$$f(\boldsymbol{x}) = -\tfrac{1}{2}\big(\tfrac{1}{4}\big) + \tfrac{1}{2}\big(\tfrac{3}{4}\big) = \boxed{\tfrac{1}{4}}, \qquad \hat{y} = +1.$$

Kernel XOR classifier

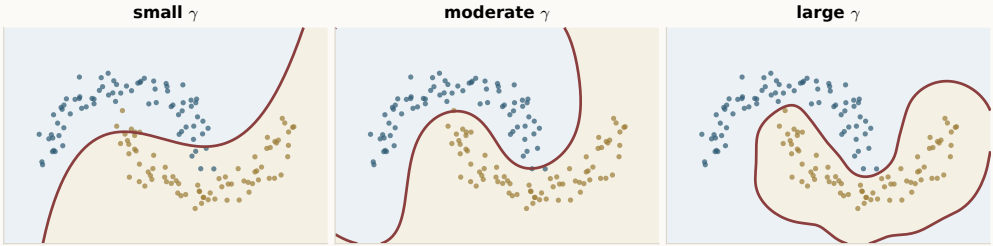
Return to XOR: $(1,1),(-1,-1)$ have label $+1$, and $(1,-1),(-1,1)$ have label -1. With $k(\boldsymbol{x},\boldsymbol{z}) = (\boldsymbol{x}^{\mathsf{T}}\boldsymbol{z})^2$, the symmetric hard-margin solution has $\alpha_i = 1/8$ and $b = 0$. For a new input $(u,v)^{\mathsf{T}}$, the decision score is

$$\begin{aligned} f(u,v) &= \tfrac{1}{8}\big[2(u+v)^2 - 2(u-v)^2\big] \\ &= \boxed{uv}. \end{aligned}$$

The nonlinear boundary $uv = 0$ separates the original XOR labels.

RBF boundaries

RBF kernels can represent more flexible boundaries; the figure shows how locality changes their shape.



Low γ gives broader influence; high γ permits more local bends.

Broad RBF influence

The smoother boundaries in the figure arise when a point influences a broad neighborhood.

$$k(\boldsymbol{x}, \boldsymbol{z}) = e^{-\gamma \|\boldsymbol{x}-\boldsymbol{z}\|_2^2}.$$

When γ is small, similarity decreases slowly with distance.

- A training point influences a broad region.
- The resulting boundary often changes smoothly.

If γ is too small, important local patterns can be missed.

Local RBF influence

More intricate boundaries become possible when each point influences only a narrow neighborhood.

- Each support vector influences a smaller region.
- The boundary can respond to finer patterns.

If γ is too large, training-specific noise may influence the boundary excessively.

Regularization and scale

Boundary shape depends on two separate choices: violation penalty and similarity scale.

Parameter	Controls	Large value
C	Margin-violation penalty	Stronger penalty
γ	RBF similarity scale	More local influence

Both affect the fitted decision boundary, so they should be chosen together using validation data.

Hinge penalty

For signed margin $m = yf(\boldsymbol{x})$, a violation contributes hinge loss:

$$\ell_{\text{hinge}} = \max(0, 1 - m).$$

If $m = 0.6$, the hinge loss is 0.4.

- With $C = 0.5$, its contribution to the objective is 0.2.
- With $C = 2$, its contribution is 0.8.

Increasing C makes the same margin violation more costly.

Distance scaling

Because RBF similarity uses Euclidean distance, raw measurement units can distort the model.

$$\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 = \sum_j (x_j - z_j)^2.$$

If one feature differs by 100 and another by 1, their contributions are 10000 and 1. Without suitable scaling, a high-range feature may dominate the kernel even when it is not the most informative.

Scaling procedure

Scaling must be fitted without information from the test set to measure generalization fairly.

1. Split the dataset before fitting preprocessing.
2. Fit the scaler on training data only.
3. Tune C and γ on validation data or by cross-validation.
4. Evaluate the final model on held-out test data.

Using test-set statistics in scaling or tuning introduces data leakage.

Computational cost

CS5487 ▪ Semester A, 2026/27

Gram-matrix memory

Implicit features reduce feature-construction work, but pairwise storage grows with sample size.

$$\boxed{\text{Memory} = O(n^2).}$$

At $n = 10{,}000$, there are 10^8 entries. Storing them as 8-byte floating-point values requires about 800 MB before other training data and overhead.

Kernel training cost

The Gram matrix is only one source of cost; training also optimizes many dual variables.

- many pairwise kernel evaluations;
- storage or repeated access to Gram entries;
- optimization over the dual coefficients α_i.

The kernel trick avoids explicit feature explosion, but does not remove dependence on sample size.

Support-vector cost

At inference time, the cost depends on the number s of retained support vectors:

$$f(\boldsymbol{x}) = \sum_{i \in SV} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b.$$

A new prediction typically needs about s kernel evaluations. For example, 500 support vectors imply roughly 500 kernel calculations per test point.

Approximate kernels

For large datasets, approximate the kernel with a finite map $\psi(\boldsymbol{x}) \in \mathbb{R}^D$:

$$\boxed{\psi(\boldsymbol{x})^\mathsf{T}\psi(\boldsymbol{z}) \approx k(\boldsymbol{x}, \boldsymbol{z}).}$$

A linear learner then operates on $\psi(\boldsymbol{x})$. Approximation trades some fidelity to the exact kernel for potentially better scalability.

Low-rank kernels

A second approximation reduces the number of values required to represent the Gram matrix.

$$\boxed{\boldsymbol{K} \approx \boldsymbol{Z}\boldsymbol{Z}^{\mathsf{T}}, \qquad \boldsymbol{Z} \in \mathbb{R}^{n \times r}.}$$

If $r \ll n$, storing $\boldsymbol{Z}$ takes $O(nr)$ rather than $O(n^2)$ entries. For $n = 10{,}000$ and $r = 100$, this is 10^6 instead of 10^8 values.

Beyond SVM

CS5487 ▪ Semester A, 2026/27

Kernel methods beyond SVM

Once an algorithm can be written using inner products, the same kernel idea may apply. Two examples:

- **Kernel K-means:** cluster by distances in feature space.
- **Kernel PCA:** find high-variance directions in feature space.

Both reuse methods introduced earlier in the course.

Feature-space centroids

K-means uses centroid distances; these distances can also be defined after a feature map. In feature space, the centroid of cluster C_k is

$$\boxed{\boldsymbol{\mu}_k^{\phi} = \frac{1}{|C_k|} \sum_{j \in C_k} \phi(\boldsymbol{x}_j).}$$

Here C_k is the set of training indices in cluster k. We can compare distances to this centroid using kernels alone.

Kernel K-means distance

Write $d_{ik}^2 = \|\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_k^\phi\|_2^2$. Expanding this distance eliminates the explicit centroid:

$$\boxed{d_{ik}^2 = k(\boldsymbol{x}_i, \boldsymbol{x}_i) - \frac{2}{|C_k|} \sum_{j \in C_k} k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{1}{|C_k|^2} \sum_{j,\ell \in C_k} k(\boldsymbol{x}_j, \boldsymbol{x}_\ell).}$$

Every term is a kernel value, so no explicit feature-space centroid is needed.

Cluster-distance example

With $k(x,z) = (1+xz)^2$, compare point $x = 2$ with the cluster $C = \{0, 1\}$.

$$k(2,2) = 25,$$
$$\sum_{z \in C} k(2,z) = 1 + 9 = 10,$$
$$\sum_{z,z' \in C} k(z,z') = 1 + 1 + 1 + 4 = 7.$$

Thus

$$\boxed{d^2 = 25 - \frac{2}{2}(10) + \frac{1}{2^2}(7) = \frac{67}{4}.}$$

Cluster assignment

For point $x = 2$, compare clusters C_1 and C_2 using their squared kernel distances.

$$C_1 = \{0, 1\}, \qquad C_2 = \{2, 3\}.$$

We found $d^2(x, C_1) = 67/4$. For C_2,

$$\begin{aligned} d^2(x, C_2) &= \tfrac{1}{4}\big[k(2,2) + k(3,3) - 2k(2,3)\big] \\ &= \tfrac{1}{4}(25 + 100 - 98) = \boxed{\tfrac{27}{4}}. \end{aligned}$$

Since $27/4 < 67/4$, kernel K-means assigns $x = 2$ to C_2.

From PCA to kernel PCA

PCA maximizes variance along linear directions in the original data. To capture nonlinear structure, apply PCA to $\phi(\boldsymbol{x})$. Kernel PCA uses feature-space inner products without explicitly constructing these new coordinates.

Kernel centering

PCA must remove the mean, and kernel PCA must do so in its implicit feature space.

$$\widetilde{\phi}_i = \phi(\boldsymbol{x}_i) - \frac{1}{n}\sum_j \phi(\boldsymbol{x}_j).$$

Let $\mathbf{1} \in \mathbb{R}^n$ contain ones and define

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T}.$$

The centered Gram matrix is

$$\boxed{\boldsymbol{K}_c = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}.}$$

Centering example

Take $x = (0, 1, 2)$ and the linear kernel $k(x, z) = xz$ to check centering numerically.

$$\boldsymbol{K} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 2 & 4 \end{bmatrix}.$$

The centered inputs are $(-1, 0, 1)$, giving

$$\boxed{\boldsymbol{K}_c = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}.}$$

This is also $\boldsymbol{HKH}$.

Kernel PCA example

Use the centered Gram matrix from the previous slide:

$$\boldsymbol{K}_c = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$

Its largest eigenvalue is 2, with unit eigenvector

$$\boxed{\boldsymbol{v} = \frac{1}{\sqrt{2}}(1, 0, -1)^{\mathsf{T}}, \qquad \boldsymbol{K}_c\boldsymbol{v} = 2\boldsymbol{v}.}$$

The eigenvector distinguishes the outer observations. Kernel PCA applies the same eigenvector calculation to nonlinear kernels.

PCA and kernel PCA

Kernel PCA changes the representation, while keeping the PCA goal of capturing variance.

	PCA	Kernel PCA
Representation	Original features	Implicit features
Main matrix	Covariance	Centered Gram matrix
Geometry	Linear	Potentially nonlinear
Operation	Eigendecomposition	Eigendecomposition

Both are unsupervised: capturing variance does not guarantee better prediction accuracy.

Model selection

CS5487 ▪ Semester A, 2026/27

Kernel selection

The kernel defines what counts as similar, so the choice should follow the data structure.

- **Linear:** original feature interactions are sufficient.
- **Polynomial:** products of features matter.
- **RBF:** nearby points should have similar behavior.
- **Specialized:** domain structure supplies a meaningful similarity.

Start with a clear baseline, then compare choices using validation results.

Parameter selection

Kernel choice alone is insufficient; the scale and regularization also affect generalization.

$$C \in \{0.1, 1, 10, 100\}, \qquad \gamma \in \{0.001, 0.01, 0.1, 1\}.$$

Each pair defines a different model. Use validation or cross-validation to select parameters; reserve the test set for final evaluation.

RBF SVM pipeline

In practice, fitting the scaler and SVM in one pipeline helps prevent data leakage.

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

model = make_pipeline(
    StandardScaler(),
    SVC(kernel="rbf", C=10.0, gamma="scale")
)
model.fit(X_train, y_train)
```

The pipeline fits scaling and SVM together on training data. Tune C and γ before final testing.

Kernel pitfalls

The main mistakes concern invalid similarities, unreliable evaluation, and excessive cost.

- Treating an arbitrary similarity as a valid kernel.
- Ignoring feature scales for RBF distances.
- Selecting C or γ using test data.
- Choosing excessive flexibility for a small dataset.
- Ignoring Gram-matrix and prediction costs.

More flexible geometry still requires careful validation.

Connections

The same approach works for several algorithms:

- **SVM:** use kernels to produce nonlinear decision boundaries.
- **K-means:** compare points with centroids in feature space.
- **PCA:** analyze variance in an implicit feature space.

$$\boxed{\text{nonlinear features} \longrightarrow \text{kernel values} \longrightarrow \text{familiar algorithms}}$$

Kernel methods: nonlinear geometry from inner products

Questions and discussion