

INSA ROUEN NORMANDIE

Machine Learning

Support Vector Machines

ITI 4 · Benoit Gaüzère

Two sessions

Linear SVM — 90 min	Kernel SVM — 90 min
Geometry and canonical margin	Explicit feature maps
Hard and soft margins	Training and prediction in feature space
Derivation of the dual	Kernel substitution in both expressions
Support vectors and prediction	Polynomial and RBF kernels
Linear prediction and synthesis	Valid kernels, model selection, SVR and novelty detection

The central question: **which computations actually require the input vectors?**

Part 1

Linear support vector machines

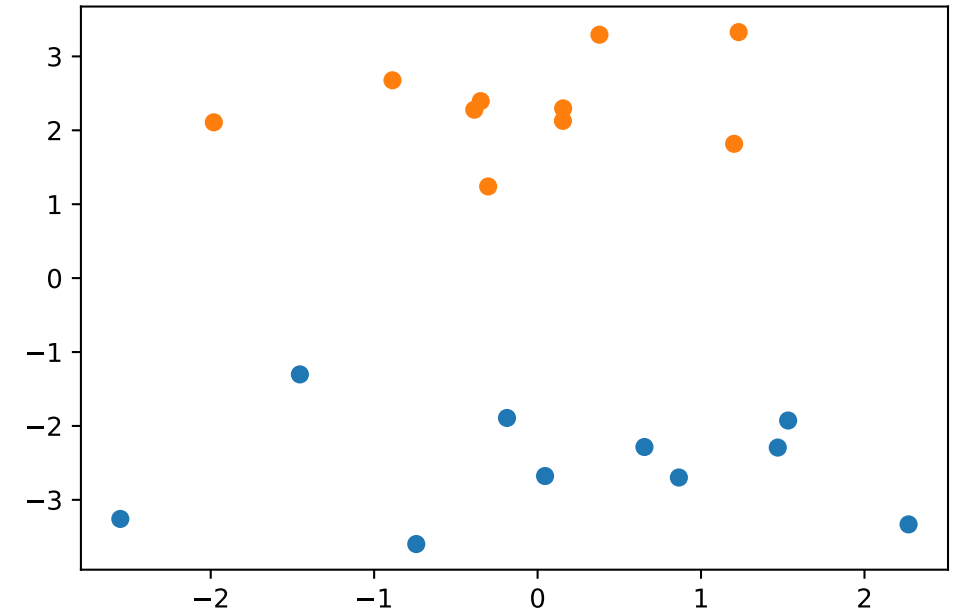
Binary classification with a linear score

Training observations: $(\mathbf{x}_i, y_i)$, with $y_i \in \{-1, +1\}$.

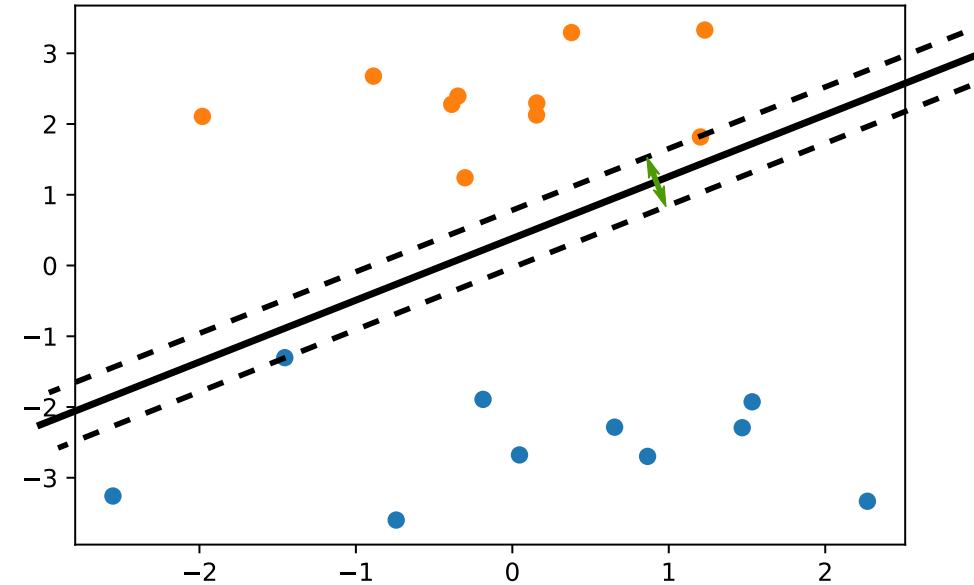
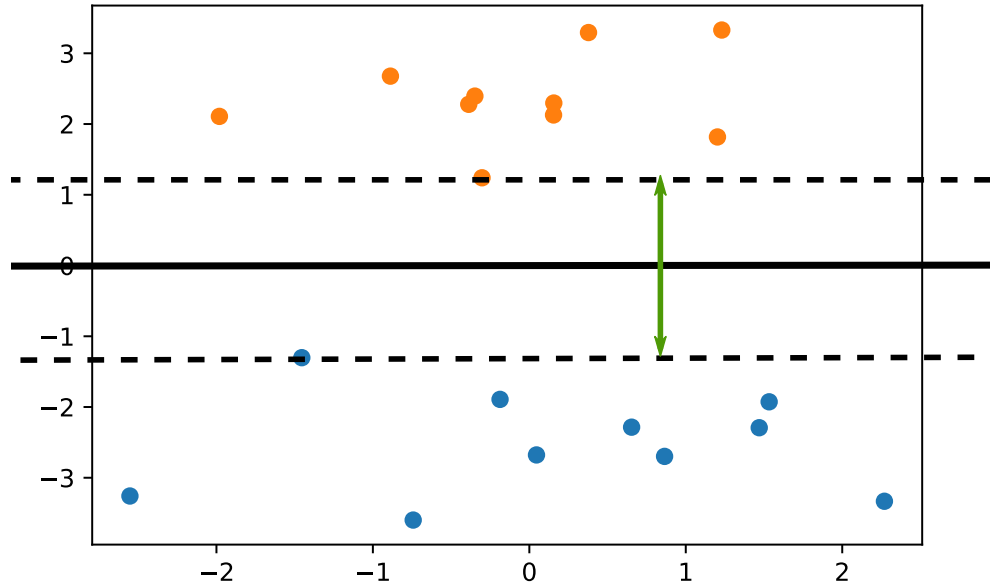
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \quad \hat{y}(\mathbf{x}) = \text{sign}(f(\mathbf{x})).$$

The separating hyperplane is $f(\mathbf{x}) = 0$.

- Correct classification: $y_i f(\mathbf{x}_i) > 0$.
- The SVM score is a real value, with zero as the default decision threshold.



Several separators fit the training labels



Both classify the observations correctly. How should we choose the boundary?

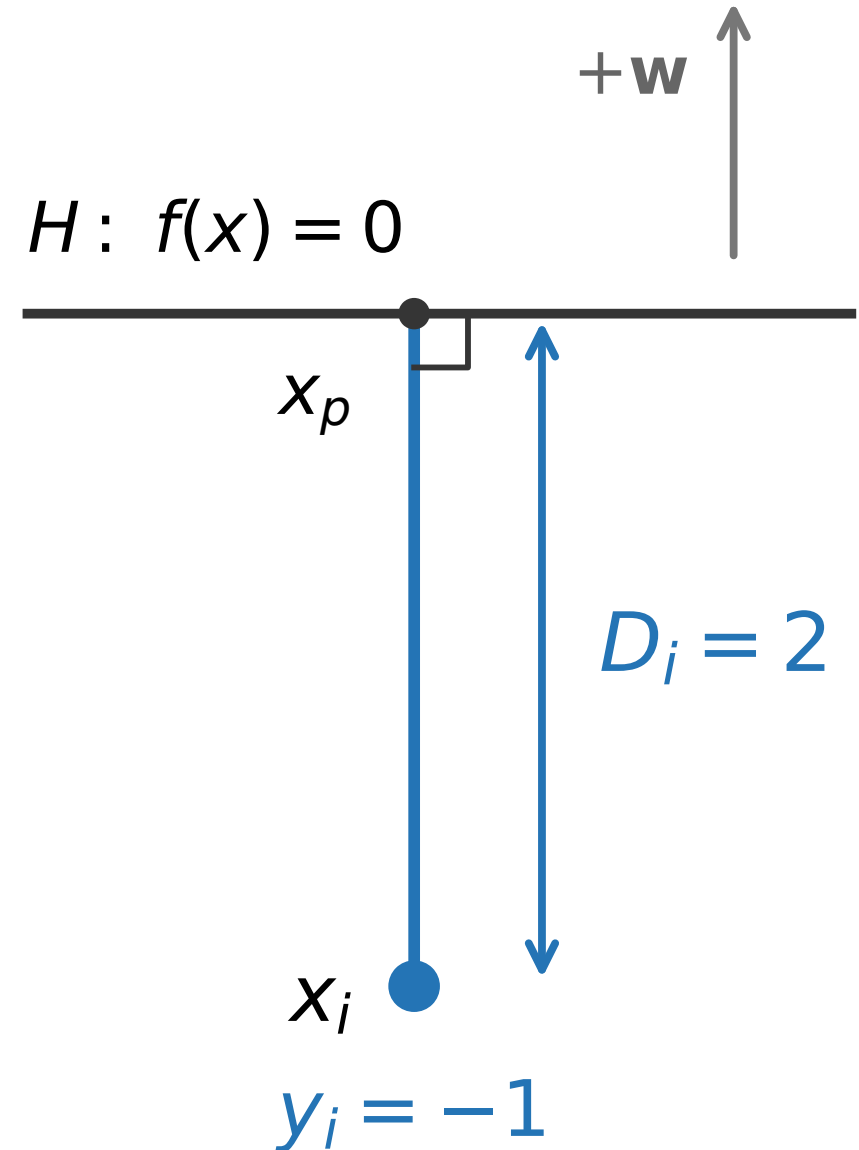
Score, distance and margins

Example: $\mathbf{w} = (0, 2)^\top$, $b = 0$, $\mathbf{x}_i = (0, -2)^\top$, $y_i = -1$.

Quantity	Definition	Example
Score	$f_i = \mathbf{w}^\top \mathbf{x}_i + b$	-4
Signed distance	$d_{s,i} = f_i / \ \mathbf{w}\ _2$	-2
Distance	$D_i = d_{s,i} $	2
Functional margin	$m_i = y_i f_i$	4
Geometric margin	$\gamma_i = y_i d_{s,i}$	2

Correct classification: $m_i, \gamma_i > 0$.

Misclassified: $m_i, \gamma_i < 0$; on the boundary: zero.



Training margin and canonical normalization

Over the training observations:

$$m = \min_i m_i, \quad \gamma = \min_i \gamma_i = \frac{m}{\|\mathbf{w}\|_2}.$$

For a strict separator, $m > 0$. Rescale **both** $\mathbf{w}$, b by $1/m$:

$$\tilde{m} = 1, \quad \boxed{\gamma = \frac{1}{\|\tilde{\mathbf{w}}\|_2}}.$$

- Rescaling leaves the hyperplane and all geometric distances unchanged.
- We fix the **functional** margin to 1; the **geometric** margin remains to be maximized.

The two margin hyperplanes

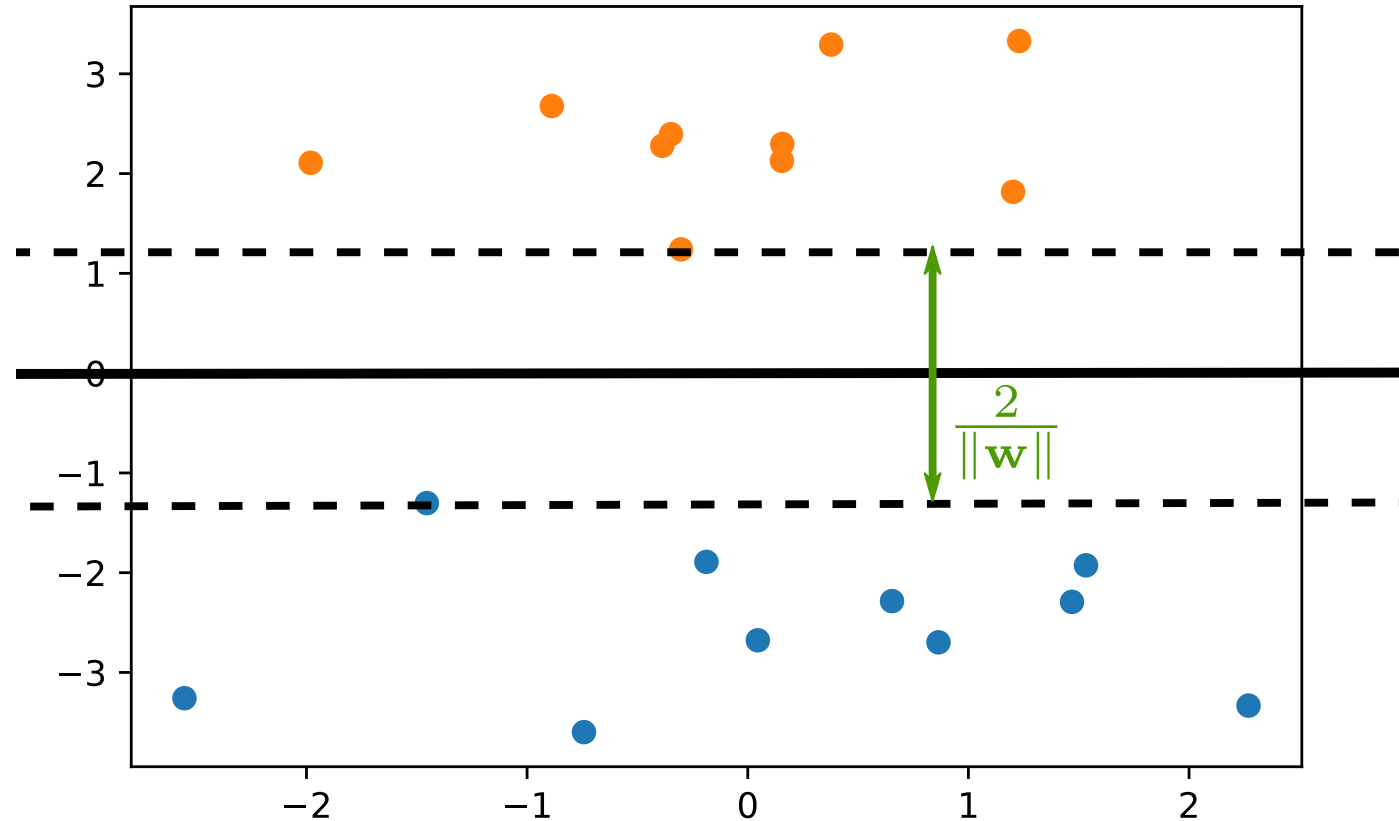
With canonically normalized parameters (tildes omitted):

$$\mathbf{w}^\top \mathbf{x} + b = +1, \quad \mathbf{w}^\top \mathbf{x} + b = -1.$$

Distance from the boundary to either plane:

$$\frac{1}{\|\mathbf{w}\|_2}.$$

Total margin width: $2/\|\mathbf{w}\|_2$.



Hard-margin optimization

Under canonical normalization, maximizing $\gamma = 1/\|\mathbf{w}\|_2$ gives

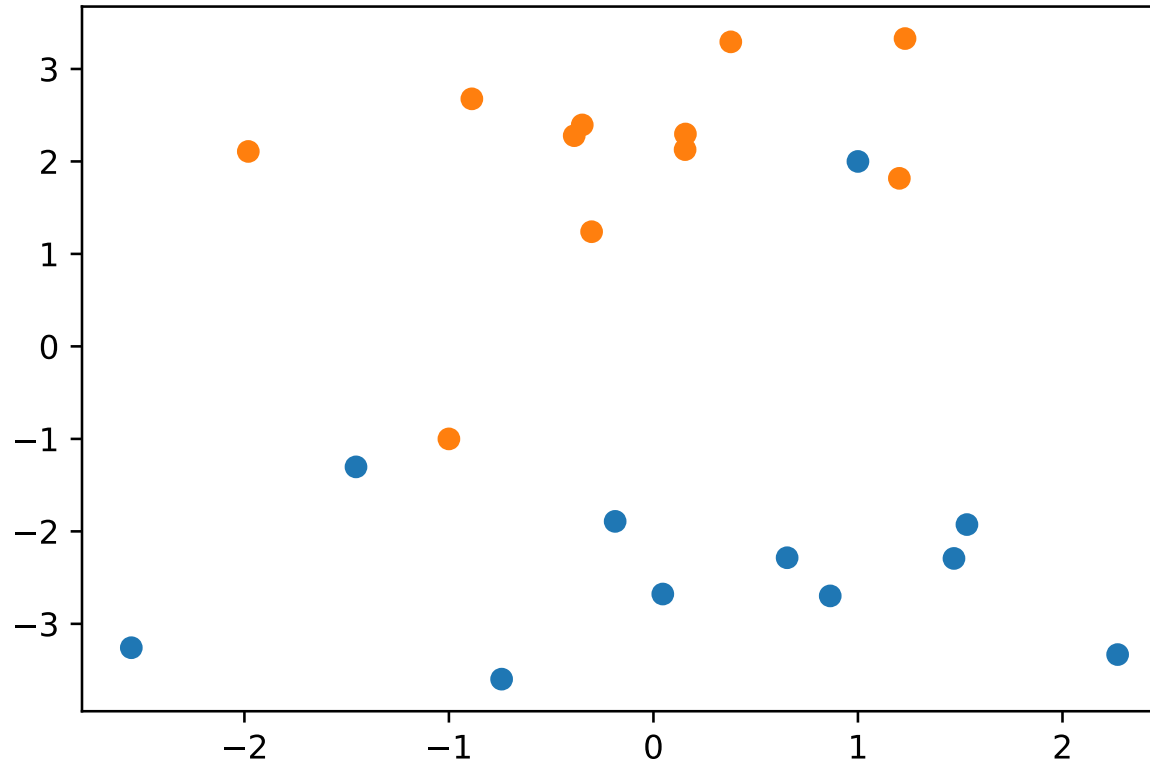
$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, n. \end{aligned}$$

The constraints impose $m_i \geq 1$. At the optimum, $\min_i m_i = 1$.

The objective selects the largest **training geometric margin**.

This convex quadratic problem is feasible only for linearly separable training data.

Overlap makes hard margin infeasible



Allow violations of the margin constraints, and penalize them.

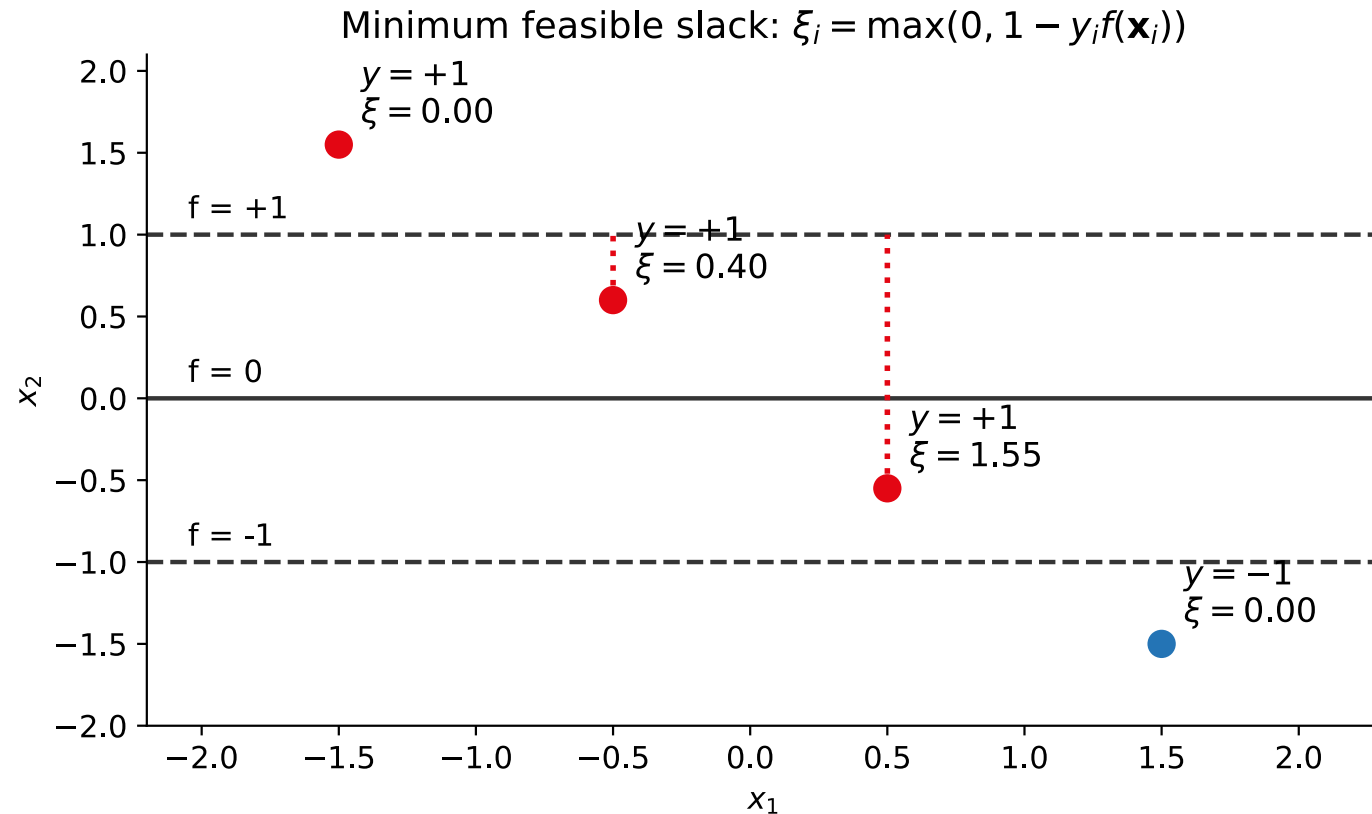
Slack variables

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Minimum feasible slack:

$$\xi_i = \max(0, 1 - y_i f(\mathbf{x}_i)).$$

- $\xi_i = 0$: on or beyond the correct margin plane.
- $0 < \xi_i < 1$: correct side, inside the margin.
- $\xi_i > 1$: wrong side of the boundary.



Soft-margin optimization

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \dots, n. \end{aligned}$$

$C > 0$ controls the penalty for margin violations.

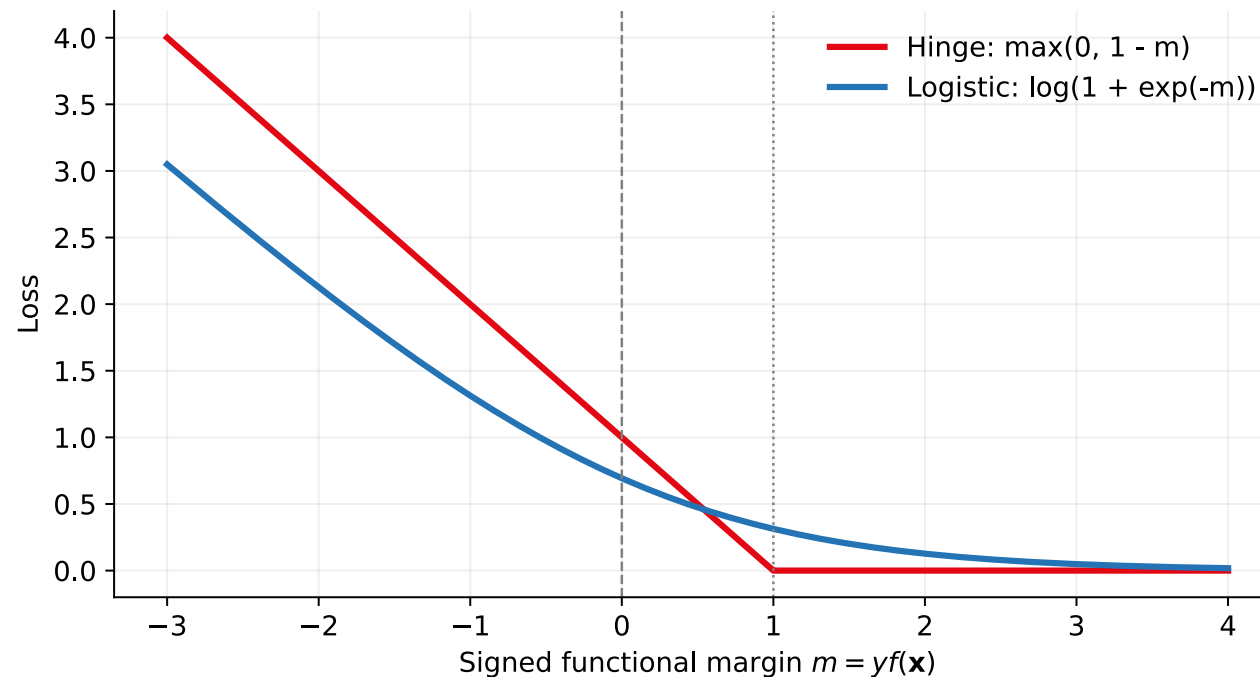
- Larger C : more weight on fitting the training observations.
- Smaller C : stronger relative regularization.

Here $1/\|\mathbf{w}\|_2$ is the band half-width; it need not equal $\min_i \gamma_i$.

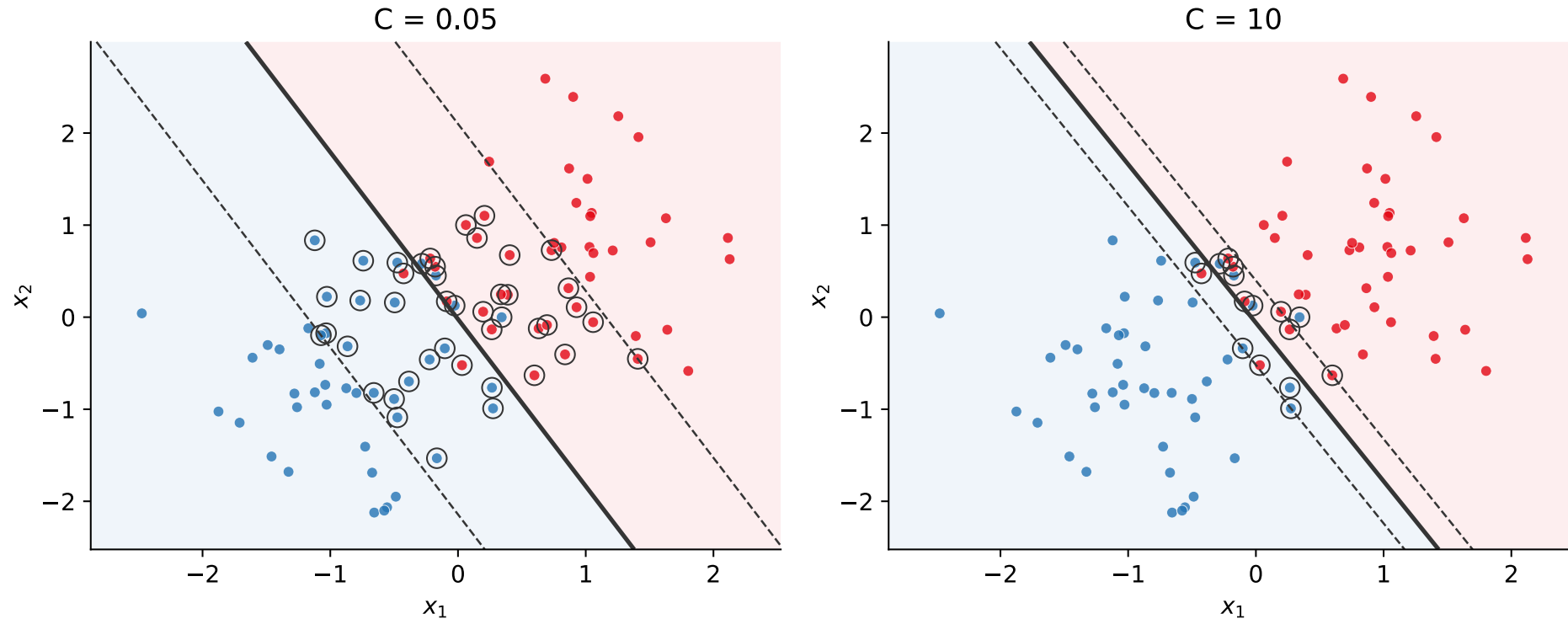
Hinge loss

Eliminating the slack variables gives

$$\min_{\mathbf{w}, b} \quad \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_i \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{hinge loss}}.$$



Effect of C on the same observations



Solid line: boundary. Dashed lines: margin planes. Circles: support vectors.

Notebook · §2 — Soft margin and hinge loss

Deriving the dual: the Lagrangian

Recall the hard-margin primal:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2, \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad \text{for all } i.$$

Write the constraints as $1 - y_i f(\mathbf{x}_i) \leq 0$.

For multipliers $\alpha_i \geq 0$,

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|_2^2 + \sum_i \alpha_i [1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)].$$

The dual maximizes the minimum of this expression over $\mathbf{w}, b$.

Eliminating the primal variables

Stationarity with respect to $\mathbf{w}$ gives

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0 \quad \Longrightarrow \quad \boxed{\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i.}$$

Stationarity with respect to b gives

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_i \alpha_i y_i = 0.$$

The fitted normal vector is a linear combination of training observations.

Substitute the minimizing normal vector

For fixed α , stationarity gives

$$\mathbf{w}_\alpha = \sum_i \alpha_i y_i \mathbf{x}_i.$$

Expand the Lagrangian before substitution:

$$\mathcal{L} = \frac{1}{2} \|\mathbf{w}\|^2 - \underbrace{\mathbf{w}^\top \sum_i \alpha_i y_i \mathbf{x}_i}_{\mathbf{w}_\alpha} - \underbrace{b \sum_i \alpha_i y_i}_0 + \sum_i \alpha_i.$$

Substitute $\mathbf{w} = \mathbf{w}_\alpha$:

$$q(\alpha) = \inf_{\mathbf{w}, b} \mathcal{L} = \frac{1}{2} \|\mathbf{w}_\alpha\|^2 - \|\mathbf{w}_\alpha\|^2 + \sum_i \alpha_i = \sum_i \alpha_i - \frac{1}{2} \|\mathbf{w}_\alpha\|^2.$$

$\mathbf{w}_\alpha$ names the minimizing $\mathbf{w}$ for these multipliers; it is not a new variable to optimize.

Substitution exposes the dot products

Expand the squared norm of $\mathbf{w}_\alpha = \sum_i \alpha_i y_i \mathbf{x}_i$:

$$\begin{aligned}\|\mathbf{w}_\alpha\|^2 &= \left(\sum_i \alpha_i y_i \mathbf{x}_i \right)^\top \left(\sum_j \alpha_j y_j \mathbf{x}_j \right) \\ &= \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j.\end{aligned}$$

Thus

$$q(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j.$$

Every dependence on the observations is now a dot product.

The hard-margin dual

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{subject to} \quad & \alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0. \end{aligned}$$

After solving for $\boldsymbol{\alpha}$, recover $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$.

Training uses the Gram matrix $G_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$.

Why support vectors appear

For $g_i = 1 - y_i f(\mathbf{x}_i) \leq 0$, KKT requires $\alpha_i g_i = 0$.

Complementary slackness for hard margin:

$$\alpha_i [y_i f(\mathbf{x}_i) - 1] = 0.$$

- If $y_i f(\mathbf{x}_i) > 1$, then $\alpha_i = 0$.
- If $\alpha_i > 0$, then $y_i f(\mathbf{x}_i) = 1$.

Only observations with $\alpha_i > 0$ contribute to $\mathbf{w}$.

These observations are the **support vectors**.

Complementary slackness: a multiplier can be positive only if its constraint is active.

Soft margin adds a box constraint

Recall the primal:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i, \quad \text{s.t.} \quad y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

With $\alpha_i \geq 0$ and $\mu_i \geq 0$:

$$\mathcal{L} = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - \xi_i - y_i f(\mathbf{x}_i)) - \sum_i \mu_i \xi_i.$$

Stationarity gives $C - \alpha_i - \mu_i = 0$, hence $0 \leq \alpha_i \leq C$.

The soft-margin dual

The dual becomes

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad \sum_i \alpha_i y_i = 0. \end{aligned}$$

Three groups in a soft-margin SVM

Dual coefficient	Consequence at the optimum
$\alpha_i = 0$	$y_i f(\mathbf{x}_i) \geq 1$: no contribution to prediction
$0 < \alpha_i < C$	$y_i f(\mathbf{x}_i) = 1$: free support vector
$\alpha_i = C$	$y_i f(\mathbf{x}_i) \leq 1$: on or inside the margin, possibly misclassified

The intercept can be recovered from any free support vector:

$$b = y_i - \mathbf{w}^\top \mathbf{x}_i, \quad 0 < \alpha_i < C.$$

Notebook · §2 — Inspect the dual coefficients and KKT conditions

Prediction also uses dot products

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^\top \mathbf{x} + b \\ &= \sum_i \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b \\ &= \sum_{i \in \mathcal{S}} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b. \end{aligned}$$

$\mathcal{S} = \{i : \alpha_i > 0\}$ is the support-vector index set.

Training needs **training–training** dot products.

Prediction needs **support–new observation** dot products.

Reconstructing a fitted linear SVM

```
from sklearn.svm import SVC

svm = SVC(kernel="linear", C=1.0).fit(X_train, y_train)
beta = svm.dual_coef_[0] # alpha_i * y_i for support vectors
support = svm.support_vectors_
w = beta @ support
b = svm.intercept_[0]

f_primal = X_new @ w + b
f_dual = (X_new @ support.T) @ beta + b
assert np.allclose(f_primal, f_dual)
assert np.allclose(f_dual, svm.decision_function(X_new))
```

Binary labels here are $-1, +1$. All arrays use the same feature scaling.

Notebook · §2 — Reconstruct prediction and the dual objective

Check-up 1: what does prediction need?

We store the support vectors, their signed coefficients $\beta_i = \alpha_i y_i$, and b .

1. Can we predict without storing $\mathbf{w}$?
2. Does every training observation appear in the prediction sum?
3. Which scalar products are needed for one new observation?

Check-up 1: answer

1.
$$f(\mathbf{x}) = \sum_{i \in \mathcal{S}} \beta_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b$$

We can predict without storing $\mathbf{w}$.

- 2. Only support vectors have nonzero coefficients in the sum.
- 3. We need one dot product between the new observation and each support vector.

Dual training and inference have different costs

For N observations in $\mathbb{R}^d$, the dual uses

$$G_{ij} = \mathbf{x}_i^\top \mathbf{x}_j, \quad G = XX^\top \in \mathbb{R}^{N \times N}.$$

- Full dense Gram: $O(N^2d)$ to compute, $O(N^2)$ to store.
- Solvers may compute entries on demand and keep a kernel cache.
- Predicting one observation needs only N_{SV} support–query products: $O(N_{\text{SV}}d)$ in the linear case.
- Compress the linear model into $\mathbf{w}, b$: prediction costs $O(d)$, with no Gram matrix.

The quadratic Gram size is not the total optimization time.

Solving the linear primal directly

Eliminate the slacks and optimize $d + 1$ parameters:

$$\min_{\mathbf{w}, b} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)).$$

- Convex, but hinge loss is not differentiable at margin 1.
- Primal methods can use subgradients or suitable coordinate updates.
- A full dense loss/subgradient pass costs $O(Nd)$; no $N \times N$ Gram is required.
- For $N \gg d$, a specialized linear solver is often attractive.

Total cost depends on iterations, tolerance, sparsity and solver.

Linear SVM: results to retain

1. Canonical normalization connects margin maximization to $\|\mathbf{w}\|_2^2$.
2. Soft margin trades regularization against hinge loss through C .
3. The fitted vector satisfies $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$.
4. Training and prediction depend on observations through dot products.
5. Support vectors have nonzero dual coefficients.

Next session: compute those dot products in a richer space.

Part 2

Nonlinear SVM and the kernel trick

Two dot-product expressions to remember

Training:

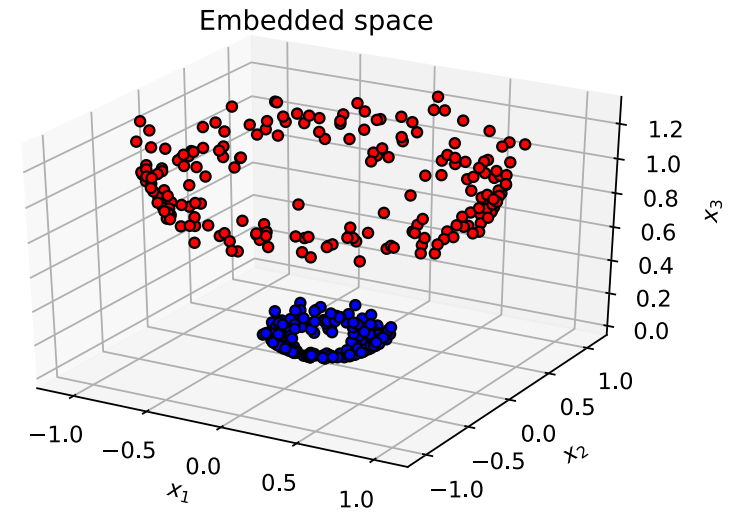
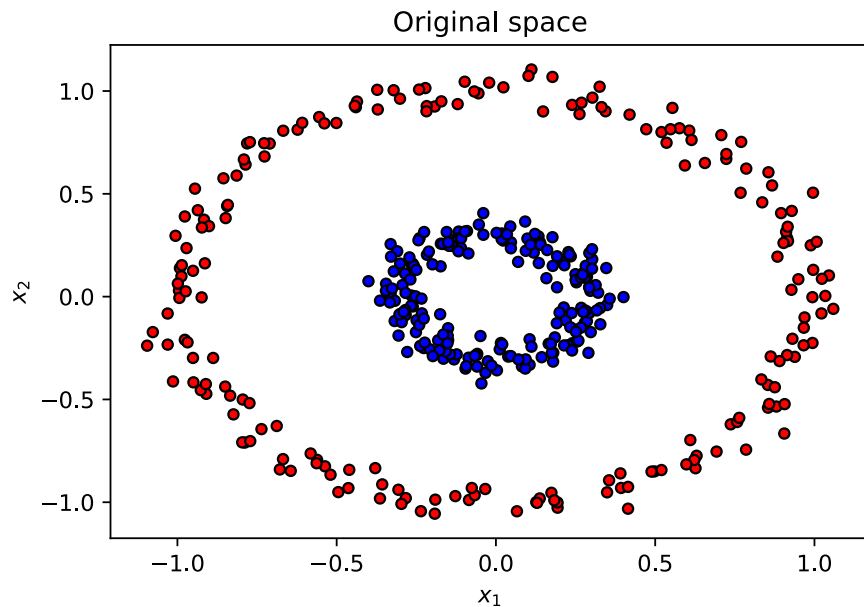
$$\max_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \quad \begin{cases} 0 \leq \alpha_i \leq C, \\ \sum_i \alpha_i y_i = 0. \end{cases}$$

Prediction:

$$f(\mathbf{x}) = \sum_{i \in \mathcal{S}} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b.$$

Can we preserve these expressions while changing the representation of the data?

A linear boundary in a richer space



$$\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2).$$

Linear in features, nonlinear in inputs

For $\phi(\mathbf{x}) = (x_1, x_2, x_1^2 + x_2^2)$, consider

$$f(\mathbf{x}) = \underbrace{(0, 0, 1)}_{\mathbf{w}_\phi^\top} \phi(\mathbf{x}) - r_0^2 = x_1^2 + x_2^2 - r_0^2.$$

The boundary is

$$x_1^2 + x_2^2 = r_0^2.$$

It is a circle in the input plane and a hyperplane in feature space.

An explicit polynomial feature map

For $\mathbf{x} = (x_1, x_2)^\top$, define

$$\phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}.$$

A linear score in this space is a quadratic function in the original coordinates.

What is $\phi(\mathbf{x})^\top \phi(\mathbf{x}')$?

Compute the product without the feature vectors

$$\begin{aligned}\phi(\mathbf{x})^\top \phi(\mathbf{x}') &= x_1^2 x_1'^2 + 2x_1 x_2 x_1' x_2' + x_2^2 x_2'^2 \\ &= (x_1 x_1' + x_2 x_2')^2 \\ &= (\mathbf{x}^\top \mathbf{x}')^2.\end{aligned}$$

Define

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}')^2.$$

We can evaluate the feature-space dot product directly from the input vectors.

Notebook · §3 — Explicit features and an implicit kernel

Check-up 2: explicit or implicit?

Let $\mathbf{x} = (1, 2)^\top$ and $\mathbf{x}' = (3, 1)^\top$.

1. Compute their degree-two feature vectors.
2. Compute $\phi(\mathbf{x})^\top \phi(\mathbf{x}')$.
3. Verify that $(\mathbf{x}^\top \mathbf{x}')^2$ gives the same value.

Check-up 2: answer

$$\phi(\mathbf{x}) = (1, 2\sqrt{2}, 4)^\top, \quad \phi(\mathbf{x}') = (9, 3\sqrt{2}, 1)^\top.$$

$$\phi(\mathbf{x})^\top \phi(\mathbf{x}') = 9 + 12 + 4 = 25.$$

$$(\mathbf{x}^\top \mathbf{x}')^2 = (3 + 2)^2 = 25.$$

The two computations are exactly equivalent.

The SVM primal in feature space

Replace each input vector by $\phi(\mathbf{x}_i)$:

$$\begin{aligned} \min_{\mathbf{w}_\phi, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}_\phi\|^2 + C \sum_i \xi_i \\ \text{subject to} \quad & y_i (\langle \mathbf{w}_\phi, \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

The model is still linear and margin-based **in feature space**.

Elimination in feature space

The same stationarity calculation gives

$$\mathbf{w}_\phi = \sum_i \alpha_i y_i \phi(\mathbf{x}_i).$$

Consequently,

$$\|\mathbf{w}_\phi\|^2 = \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle,$$

$$f(\mathbf{x}) = \sum_i \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b.$$

Both computations need only inner products of feature vectors.

Kernel trick: the training objective

Define $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$.

The feature-space dual is

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

subject to

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

Evaluate the kernel matrix without constructing $\phi(\mathbf{x}_i)$ explicitly.

Kernel trick: prediction

After fitting the kernel SVM,

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

For a free support vector, the intercept satisfies

$$b = y_i - \sum_{j \in \mathcal{S}} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i).$$

Training and prediction use the **same kernel**.

Kernel matrices: training and prediction

Let n be the number of training observations and m the number of new observations.

Computation	Entries	Shape
Training Gram matrix	$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$	$n \times n$
Prediction matrix	$K_{ri}^{\text{new}} = k(\mathbf{x}_r^{\text{new}}, \mathbf{x}_i)$	$m \times n$
Prediction using supports only	same entries for $i \in \mathcal{S}$	$m \times n_{\text{SV}}$

With $\beta_i = \alpha_i y_i$,

$$\mathbf{f}_{\text{new}} = K^{\text{new}} \boldsymbol{\beta} + b \mathbf{1}.$$

Reconstructing kernel SVM predictions

```
from sklearn.metrics.pairwise import rbf_kernel
from sklearn.svm import SVC

svm = SVC(kernel="rbf", C=1.0, gamma=0.5)
svm.fit(X_train, y_train)

K_new = rbf_kernel(X_new, svm.support_vectors_, gamma=0.5)
f_manual = K_new @ svm.dual_coef_[0] + svm.intercept_[0]
assert np.allclose(f_manual, svm.decision_function(X_new))
```

This binary example computes the same scores without an explicit feature map.

Notebook · §4 — Kernel prediction: support–new products

What makes a kernel valid?

For every finite set of observations, the Gram matrix must be symmetric and **positive semidefinite**:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \quad \mathbf{c}^\top K \mathbf{c} \geq 0 \quad \text{for all } \mathbf{c}.$$

For a feature-space inner product,

$$\mathbf{c}^\top K \mathbf{c} = \left\| \sum_i c_i \phi(\mathbf{x}_i) \right\|^2 \geq 0.$$

This property also keeps the SVM dual objective concave.

Notebook · §3 — Validity: positive entries are not enough

Check-up 3: is similarity enough?

Suppose a symmetric similarity gives, on two observations,

$$K = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

All entries are positive. Is this sufficient for a valid kernel?

Evaluate $\mathbf{c}^\top K \mathbf{c}$ for $\mathbf{c} = (1, -1)^\top$.

Check-up 3: answer

$$\mathbf{c}^\top K \mathbf{c} = -2 < 0.$$

The matrix has eigenvalues 3 and -1 .

Positive entries and symmetry do not ensure positive semidefiniteness.

This similarity cannot be an inner-product kernel on these observations.

Common kernels on vectors

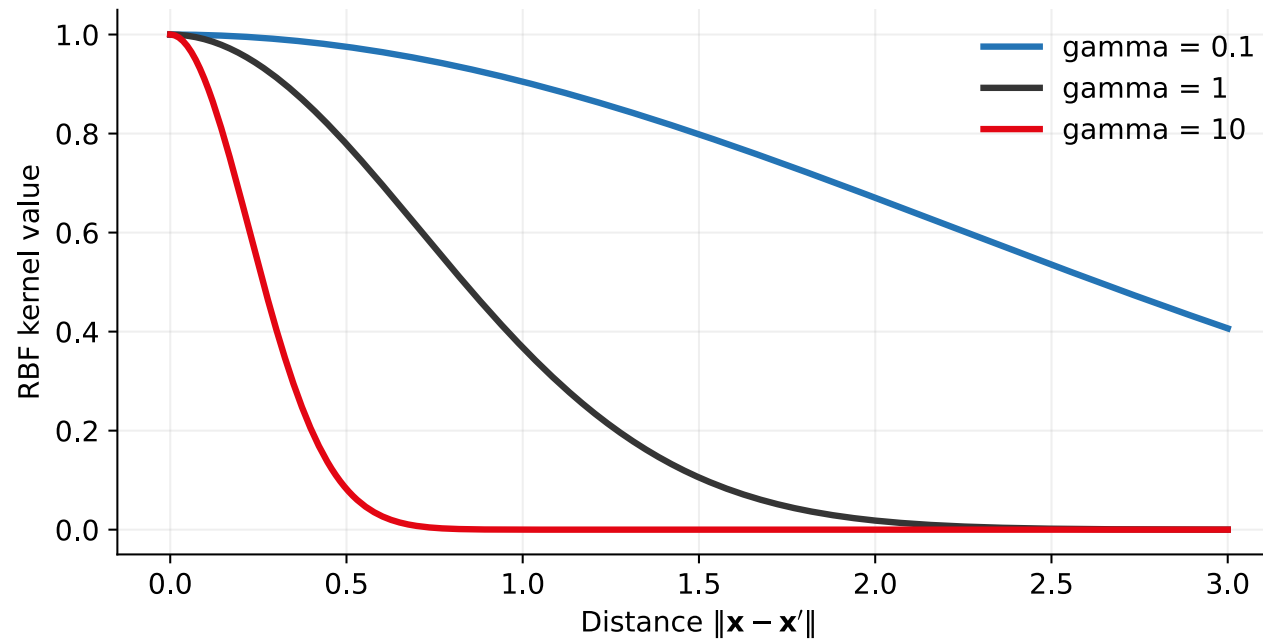
Kernel	Definition	Parameters
Linear	$\mathbf{x}^\top \mathbf{x}'$	none
Polynomial	$(\gamma \mathbf{x}^\top \mathbf{x}' + c_0)^p$	degree $p \in \mathbb{N}^*$, $\gamma > 0$, $c_0 \geq 0$
Gaussian / RBF	$\exp(-\gamma \ \mathbf{x} - \mathbf{x}'\ _2^2)$	$\gamma > 0$

The degree-two example used $p = 2$, $\gamma = 1$, $c_0 = 0$.

For each kernel, C still controls the cost of margin violations.

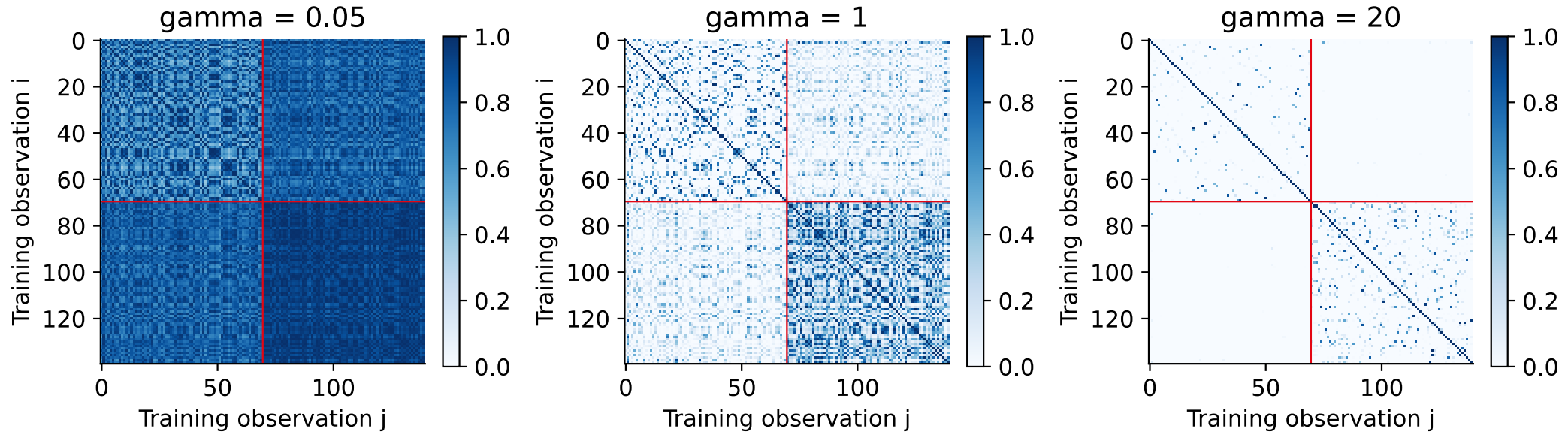
RBF: gamma sets a length scale

$$k(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2}, \quad \gamma = \frac{1}{2\sigma^2}.$$



Larger γ gives a narrower kernel. Smaller γ gives a broader kernel.

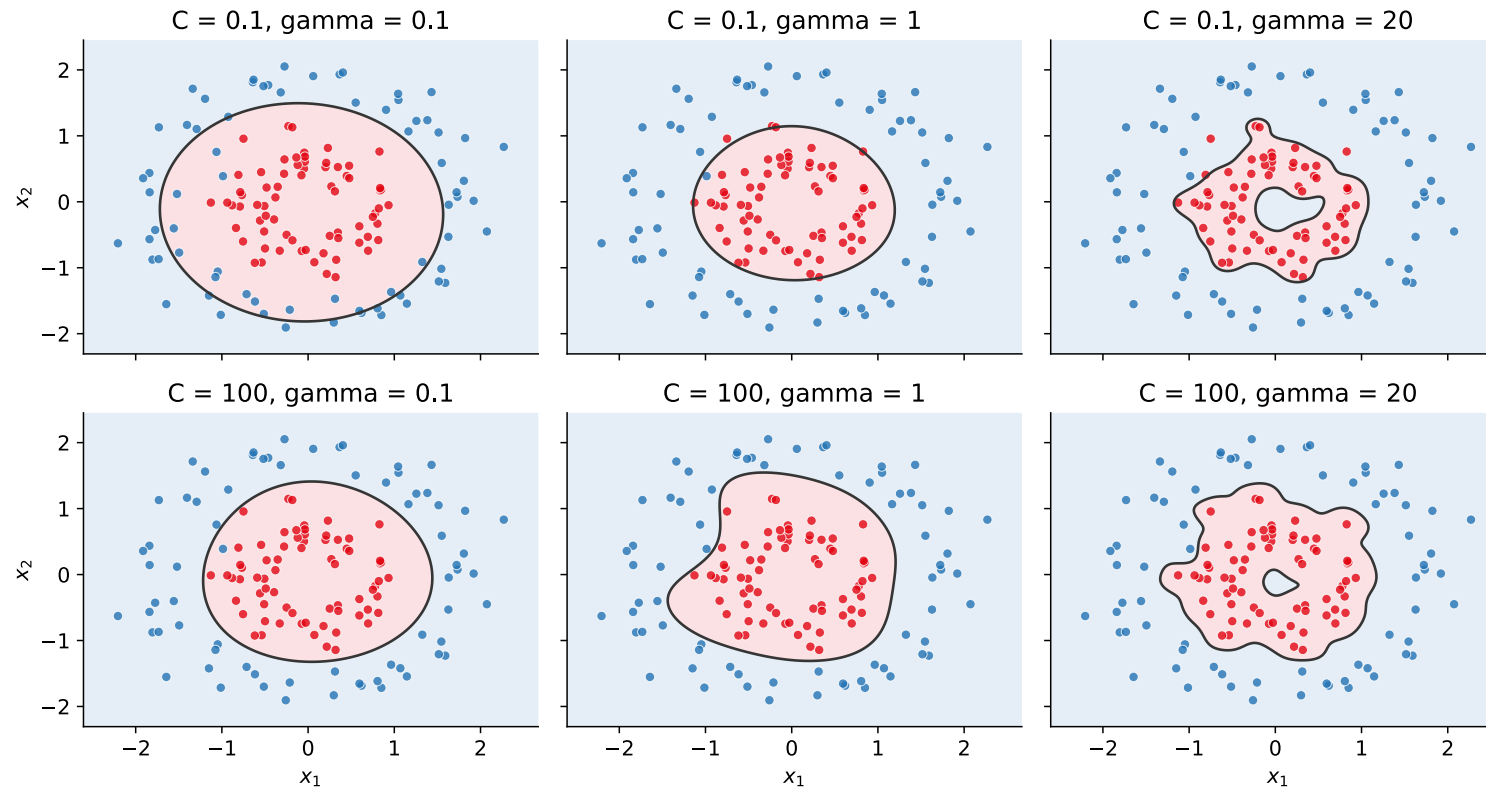
Gamma also changes the Gram matrix



Observations are ordered by class. The red lines mark the class boundary in this ordering.

Small γ : many similarities are high. Large γ : similarities become local.

C and gamma interact



Same training observations. Rows vary C ; columns vary γ .

Notebook · §4 — RBF: gamma changes the representation

Select the whole pipeline

```
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

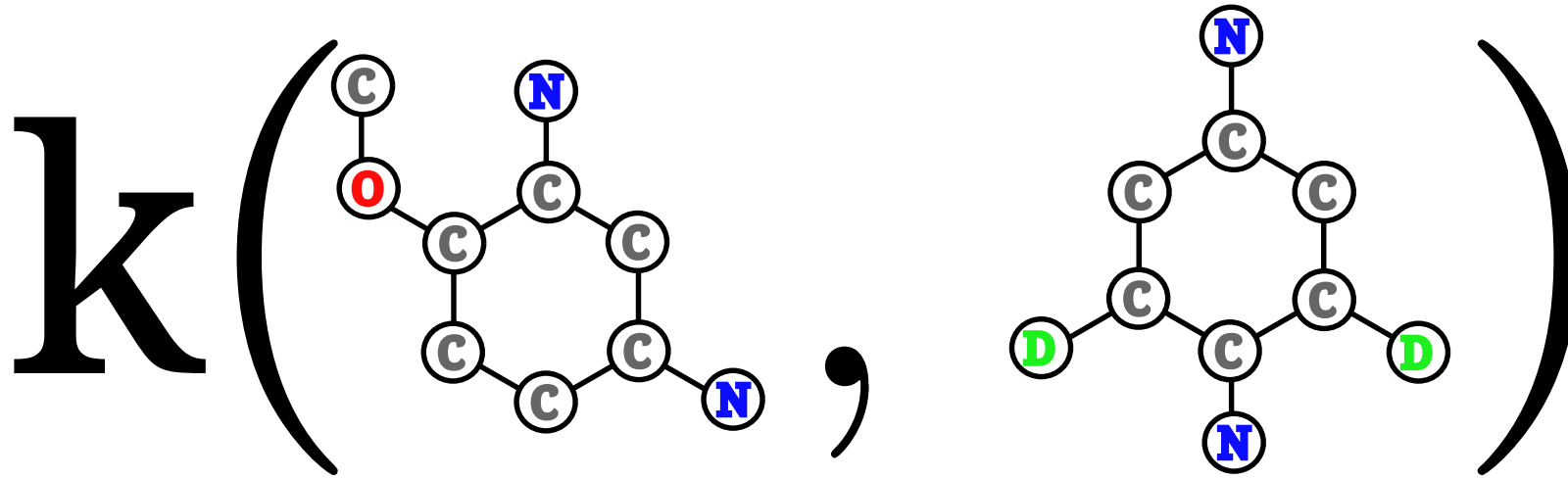
pipeline = Pipeline([
    ("scale", StandardScaler()),
    ("svm", SVC(kernel="rbf")),
])
cv = StratifiedKFold(5, shuffle=True, random_state=42)
search = GridSearchCV(pipeline, {
    "svm__C": [0.1, 1, 10, 100],
    "svm__gamma": [0.01, 0.1, 1, 10],
}, cv=cv, scoring="roc_auc")
search.fit(X_dev, y_dev)
```

Scores, decisions and computational cost

- Binary SVM: use $f(\mathbf{x})$ for ranking and ROC, then a threshold for decisions.
- A kernel matrix contains n^2 entries if stored densely.
- Kernel prediction uses n_{SV} kernel evaluations per observation.
- A linear SVM can compress the support-vector sum into $\mathbf{w}$.

Large numbers of observations or support vectors can make kernel SVM costly.

Kernels on graphs



Describe a graph G by counts of selected patterns:

$$\phi(G) = (c_1(G), \dots, c_q(G))^{\top}, \quad k(G, G') = \sum_{j=1}^q c_j(G)c_j(G').$$

The kernel SVM equations apply to these graphs as well.

Building kernels from known ones

If k_1 and k_2 are valid kernels, then so are

$$ak_1 + bk_2 \quad (a, b \geq 0), \quad k_1 k_2.$$

For nonzero feature vectors, normalization gives

$$\tilde{k}(x, x') = \frac{k(x, x')}{\sqrt{k(x, x)k(x', x')}}.$$

For graph counts, normalization reduces the effect of overall graph size.

Support vector regression: slack variables

The target y_i is real-valued. Let $f_i = \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b$.

$$\begin{aligned} \min_{\mathbf{w}, b, \xi, \xi^*} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i (\xi_i + \xi_i^*) \\ \text{subject to} \quad & y_i - f_i \leq \varepsilon + \xi_i, \\ & f_i - y_i \leq \varepsilon + \xi_i^*, \quad \xi_i, \xi_i^* \geq 0. \end{aligned}$$

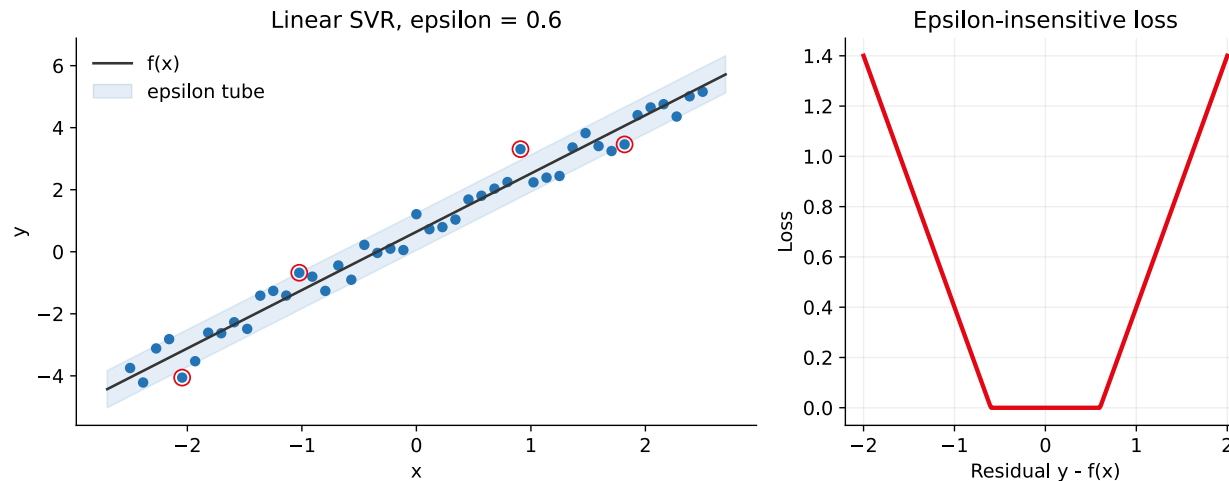
Two slacks per observation: one for each side of the tube.

SVR: epsilon-insensitive loss

At fixed f_i , the minimum slacks satisfy

$$\xi_i = \max(0, y_i - f_i - \varepsilon), \quad \xi_i^* = \max(0, f_i - y_i - \varepsilon).$$

Thus $\xi_i + \xi_i^* = \max(0, |y_i - f_i| - \varepsilon)$.



No penalty inside the tube. Linear penalty beyond its edges.

SVR: objective and kernel prediction

$$\min_{\mathbf{w}, b} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \max(0, |y_i - f_i| - \varepsilon).$$

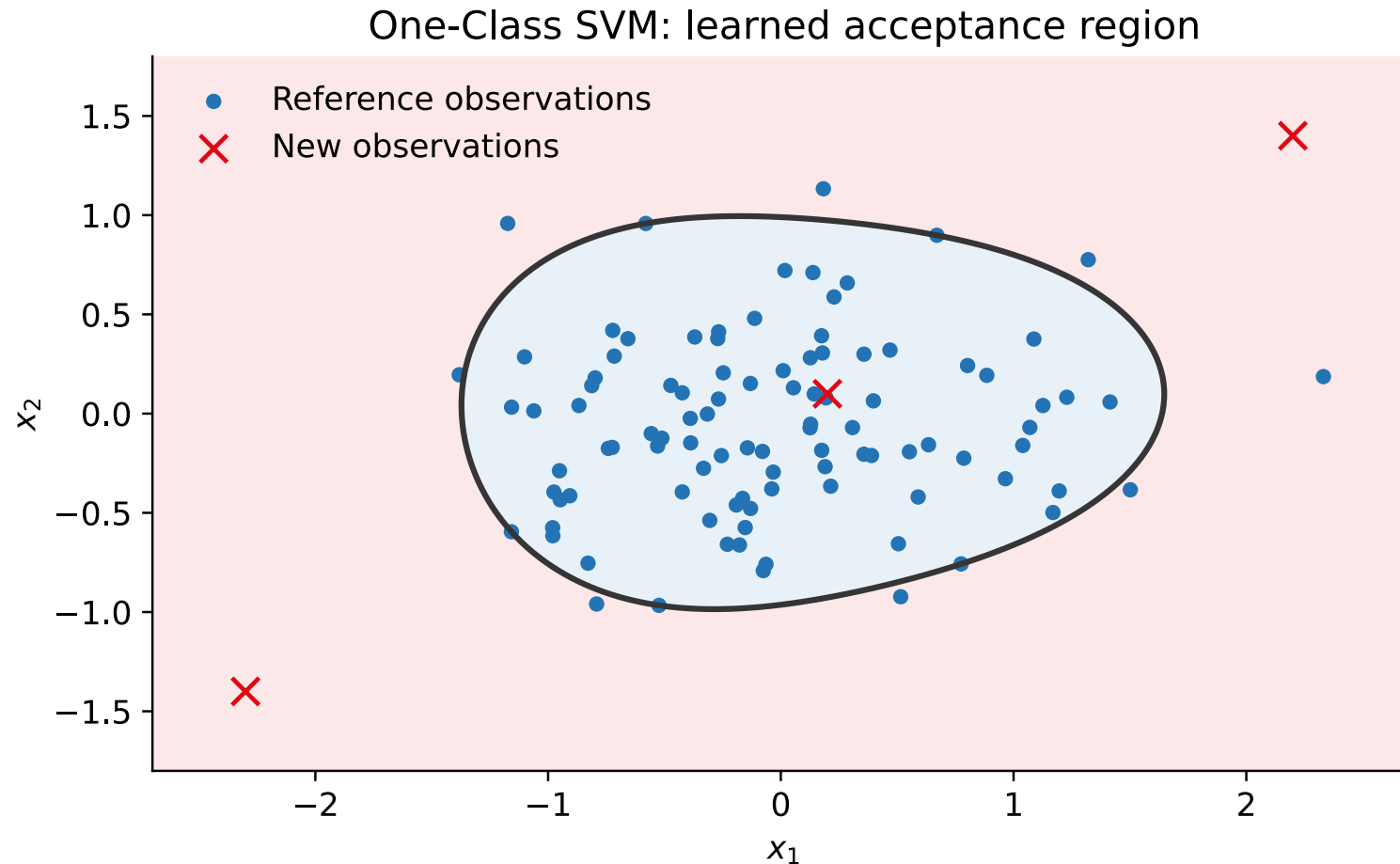
- C controls the penalty outside the tube.
- ε sets the tolerance in target units.

The fitted predictor also uses the kernel:

$$f(\mathbf{x}) = \sum_i (\alpha_i - \alpha_i^*) k(\mathbf{x}_i, \mathbf{x}) + b.$$

One-Class SVM: novelty detection

Fit a model to reference observations, usually representing normal behaviour.



One-Class SVM: optimization

For n reference observations and $\nu \in (0, 1]$:

$$\begin{aligned} \min_{\mathbf{w}, \rho, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu n} \sum_i \xi_i - \rho \\ \text{subject to} \quad & \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle \geq \rho - \xi_i, \quad \xi_i \geq 0. \\ & f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho. \end{aligned}$$

Accept if $f(\mathbf{x}) \geq 0$; flag as unusual otherwise.

One-Class SVM in practice

```
from sklearn.svm import OneClassSVM

detector = Pipeline([
    ("scale", StandardScaler()),
    ("svm", OneClassSVM(kernel="rbf", gamma=0.5, nu=0.1)),
])
detector.fit(X_reference)
scores = detector.decision_function(X_new)
flags = detector.predict(X_new)  # +1 = accepted, -1 = unusual
```

$\nu \in (0, 1]$ controls the trade-off. It is not an exact anomaly-rate setting.

Notebook · §6 — One-Class SVM: learn a reference region

Three SVM tasks

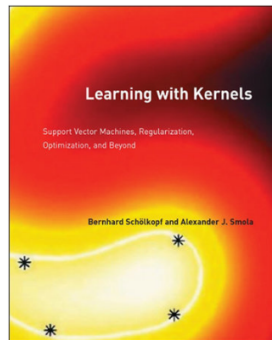
Task	Training information	Model output	Main parameters
Classification	binary or multiclass labels	class score and decision	C , kernel parameters
Regression	real-valued targets	real-valued prediction	C , ε , kernel parameters
Novelty detection	reference observations	acceptance score and flag	ν , kernel parameters

The kernel representation extends the input space in all three cases.

Kernel SVM: key points and further reading

1. A feature map changes the representation; the SVM remains linear in that space.
2. Training and prediction both use feature-space inner products.
3. A kernel computes them without constructing the feature vectors.
4. Valid kernels have symmetric positive-semidefinite Gram matrices.
5. Select C and kernel parameters together on development data.

Going further



Adaptive Computation And Machine Learning Series

Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond



By Bernhard Schölkopf, Alexander J. Smola