AI Research Labs

Section 6 of 6 · 7 lessons

Clustering and Support Vector Machines

How K-means groups observations without labels, and how SVMs build the widest-margin decision boundary.

6.1 K-means Clustering

6.2 Distance Calculations and Cluster Assignment

6.3 Updating Centroids and Convergence

6.4 SVM Fundamentals

6.5 Margins and Support Vectors

6.6 Hard-margin and Soft-margin SVM

6.7 The C Parameter and Kernels



Free course with quizzes, progress tracking and Scoopy, an AI tutor:
airesearchlabs.si/learn/ml

Machine Learning Foundations · Edition November 2026 · © 2026 AI Research Labs. Free for personal study.

Where this section fits

Every lesson is a step in building a model. These are the steps this section teaches; the Big picture PDF and the online map show how they connect to the rest.

C1

SUPERVISED LEARNING · CLASSIFICATION LANE · STEP C1

Choose the model

How do I turn inputs into a class, or a probability of one?

Lessons here: 6.4, 6.5 · feeds into Turn predictions into decisions

C2

SUPERVISED LEARNING · CLASSIFICATION LANE · STEP C2

Fit it

Which coefficients make the observed labels most probable?

Lessons here: 6.6 · feeds into Control complexity

5

SUPERVISED LEARNING · STEP 5

Control complexity

How simple can the model be and still capture the pattern?

Lessons here: 6.7 · feeds into Turn predictions into decisions

1

UNSUPERVISED LEARNING · STEP 1

Measure similarity

What does it mean for two customers to be "close"?

Lessons here: 6.2

2

UNSUPERVISED LEARNING · STEP 2

Assign and update

How does K-means find the groups?

Lessons here: 6.1

3

UNSUPERVISED LEARNING · STEP 3

Choose K and judge the result

How many groups, and are they any use?

Lessons here: 6.3 · feeds into Control complexity, Measure similarity

K-means Clustering

K-means finds K groups in unlabelled data by alternating two simple steps. What clustering is for, the algorithm in four steps, what it minimises, and how it differs from K-nearest neighbours.

Clustering

So far every model had a label Y to predict. **Clustering** has no labels: it partitions observations into groups so that points in the same cluster are similar and points in different clusters are different. Typical uses: customer segmentation, grouping patients with similar profiles, compressing data into K prototypes, exploratory analysis.

Because there's no "right answer" column, there's no test-set accuracy. Quality is judged by internal measures (how tight the clusters are) and by usefulness.



Before: unlabelled points.



After K-means, K = 3: each point joins its nearest centroid (×).

The algorithm (Lloyd's algorithm)

Input: data $x_1, \dots, x_n \in \mathbb{R}^p$ and the number of clusters K. μ_k is the centroid (mean) of cluster k.

1. **Initialise:** pick K starting centroids, e.g. K random data points (or the smarter k-means++).
2. **Assignment step:** assign every point to its nearest centroid: $c_i = \arg\min_k \|x_i - \mu_k\|^2$.
3. **Update step:** move each centroid to the mean of its assigned points: $\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$.
4. **Repeat** steps 2–3 until the assignments stop changing.

TWO ALTERNATING GUESSES

Step 2 asks "given where the centres are, who belongs to whom?" Step 3 asks "given who belongs to whom, where is the best centre?" Each step can only improve the objective, so the process settles down.

What it minimises

WITHIN-CLUSTER SUM OF SQUARES (WCSS, INERTIA)

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2$$

The total squared distance from each point to its own cluster's centre.

It's a squared-error loss with no labels: each point's "target" is its own centroid. The algorithm finds a **local** minimum of J, not necessarily the global one (6.3).

K-means vs K-nearest neighbours

	K-MEANS	K-NEAREST NEIGHBOURS
Learning type	Unsupervised (no labels)	Supervised (labels needed)
Task	Clustering	Regression or classification
What K is	Number of clusters	Number of neighbours used to predict
Output	Cluster assignments + centroids	Predicted value or class

WRONG TOOL

"K-means is used to predict whether a patient will be readmitted." If labels exist and prediction is the goal, use supervised learning. Clusters can be useful *features*, but they aren't predictions.

Cheatsheet

K-means

1) Initialise K centroids. 2) Assign each point to the nearest centroid. 3) Move each centroid to its cluster mean. 4) Repeat until assignments stop changing.

K-means objective

$J = \sum_k \sum_{i \in C_k} \|x_i - \mu_k\|^2$ (WCSS). Local minimum only.

K-means vs K-NN

K-means: unsupervised, K = clusters. K-NN: supervised, K = neighbours.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/clustering-and-svm/k-means-clustering

Distance Calculations and Cluster Assignment

Euclidean and Manhattan distance, how to assign points to centroids by hand, a fully worked 2-D example, and why unscaled features let one variable dominate the clusters.

Measuring distance

DISTANCES BETWEEN A AND B IN P DIMENSIONS

$$\text{Euclidean: } \|a-b\| = \sqrt{\sum_{j=1}^{p}(a_j-b_j)^2} \qquad \text{Manhattan: } \sum_{j=1}^{p}|a_j-b_j|$$

K-means uses (squared) Euclidean distance. For assignment, comparing squared distances gives the same answer as comparing distances, and skips the square root.

QUICK CHECK

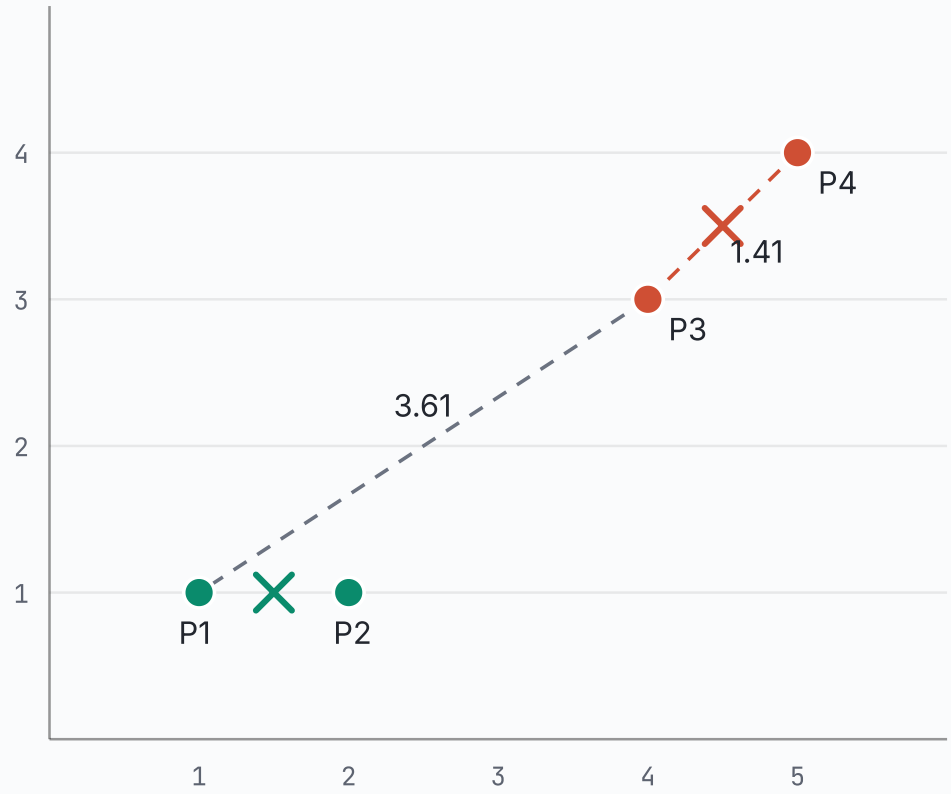
$a=(1,1)$, $b=(4,5)$: Euclidean $=\sqrt{9+16}=5$; squared $=25$; Manhattan $=3+4=7$.

Worked example: one assignment step

Points $P_1(1,1)$, $P_2(2,1)$, $P_3(4,3)$, $P_4(5,4)$; $K=2$; initial centroids $\mu_1=P_1=(1,1)$ and $\mu_2=P_4=(5,4)$.

POINT	DISTANCE TO $M_1(1,1)$	DISTANCE TO $M_2(5,4)$	ASSIGN
$P_1(1,1)$	0	$\sqrt{16+9}=5$	C_1
$P_2(2,1)$	$\sqrt{1+0}=1$	$\sqrt{9+9}=4.24$	C_1
$P_3(4,3)$	$\sqrt{9+4}=3.61$	$\sqrt{1+1}=1.41$	C_2
$P_4(5,4)$	5	0	C_2

So $C_1=\{P_1,P_2\}$ and $C_2=\{P_3,P_4\}$. The update step comes next (6.3).



P3 is 1.41 from μ_2 but 3.61 from μ_1, so it joins cluster 2. Crosses show where the centroids move after the update step.

Scale matters

Distance adds up squared differences across features, so a feature with a large numerical range dominates. Cluster customers on annual income (£20,000–£200,000) and store visits (1–40), and the clusters will split purely by income: a £1,000 income difference outweighs a 30-visit difference.

FIX

Standardise features (mean 0, SD 1) before K-means, unless the raw units are genuinely comparable. The same applies to K-NN and SVMs.

CATEGORIES

Means and Euclidean distance are undefined for categorical data ("red", "green"). Use variants such as k-modes or a suitable distance.

Cheatsheet

Distances

Euclidean $\sqrt{\sum(a_j - b_j)^2}$; Manhattan $\sum |a_j - b_j|$. K-means compares squared Euclidean.

Unscaled features

The largest-range feature dominates distances. Standardise before K-means, K-NN, SVM.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/clustering-and-svm/distance-calculations-and-cluster-assignment

Updating Centroids and Convergence

The update step, why K-means always converges but only to a local optimum, the effect of a bad start, choosing K with the elbow method, and the main ways K-means goes wrong.

The update step, continued

From 6.2: $C_1 = \{P_1(1,1), P_2(2,1)\}$, $C_2 = \{P_3(4,3), P_4(5,4)\}$. Move each centroid to its cluster mean:

$$\mu_1 = \left(\tfrac{1+2}{2}, \tfrac{1+1}{2}\right) = (1.5, 1), \qquad \mu_2 = \left(\tfrac{4+5}{2}, \tfrac{3+4}{2}\right) = (4.5, 3.5)$$

Iteration 2: reassign. Every point is still closest to the same centroid, so assignments don't change: **converged**.

$J = (0.25 + 0.25) + (0.5 + 0.5) = 1.5$.

Why it converges

- The assignment step can't increase J: each point moves to a closer (or equally close) centre.
- The update step can't increase J: the mean minimises the sum of squared distances within a cluster.
- There are finitely many ways to assign points, so J can't keep decreasing forever.

But it converges to a **local** minimum, which depends on the starting centroids.

WORKED EXAMPLE: A POOR START THAT RECOVERS

1-D points $1, 2, 4, 11, 12, 14$; $K = 2$; start at $\mu_1 = 1$, $\mu_2 = 2$.

1. Assign: 1 → C_1; 2, 4, 11, 12, 14 → C_2. Update: $\mu_1 = 1$, $\mu_2 = 43/5 = 8.6$.
2. Assign: 1, 2 (1 vs 6.6), 4 (3 vs 4.6) → C_1; 11, 12, 14 → C_2. Update: $\mu_1 = 7/3 = 2.33$, $\mu_2 = 37/3 = 12.33$.
3. Assign: no change. Converged. $J = (1.78 + 0.11 + 2.78) + (1.78 + 0.11 + 2.78) = 9.33$.

Here a bad start still found the natural groups. On other data it can get stuck in a poor solution, so K-means is run **several times from different starts** and the run with the lowest J is kept. k-means++ chooses spread-out starting centroids to make bad starts rarer.

Choosing K in K-means

J always decreases as K grows (with $K = n$, every point is its own cluster and $J = 0$), so you can't choose K by minimising J, just as training error can't choose model complexity.



Elbow method: big drops up to K = 3, then only small gains. Choose the elbow.

- **Elbow method:** plot J against K and choose the bend where extra clusters stop helping much.
- **Silhouette score:** for each point, $s = \frac{b-a}{\max(a,b)} \in [-1, 1]$, where a is the mean distance to its own cluster and b to the nearest other cluster. Higher average is better.
- **Usefulness and stability:** are the clusters actionable, and do they stay similar on resampled data?

When K-means goes wrong

ISSUE	WHY	WHAT TO DO
Features on different scales	Large-scale variable dominates distance	Standardise first
Sensitive to initialisation	Only a local optimum	Many restarts; k-means++; keep lowest J
K must be chosen	Not learned	Elbow, silhouette, domain use
Assumes round, similar-sized clusters	Euclidean distance to a mean	Other methods (mixtures, density-based) for elongated or nested clusters
Outliers	Means are pulled by extremes	Handle outliers; consider k-medians
Hard assignments only	Each point fully in one cluster	Soft clustering (mixture models)

Cheatsheet

Choosing K

Not by min J (falls with K). Use the elbow, silhouette, stability and usefulness.

K-means failure modes

Unscaled features · bad starts (local optima) · wrong K · non-round clusters · outliers · categorical data.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/clustering-and-svm/updating-centroids-and-convergence

SVM Fundamentals

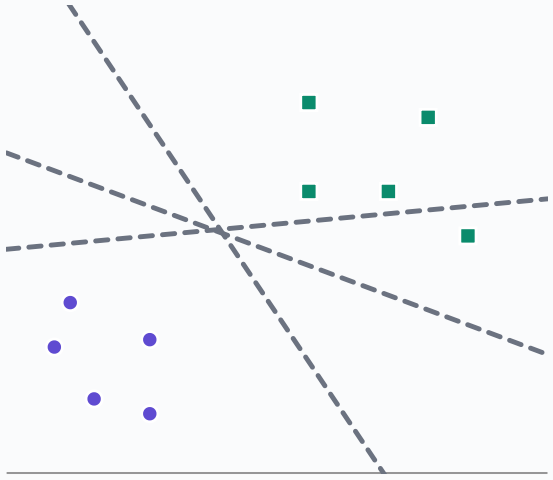
A support vector machine is a linear classifier that chooses the separating line with the widest safety margin. The idea, the ±1 label convention, the decision rule, and the link to regularisation.

The idea: the widest street

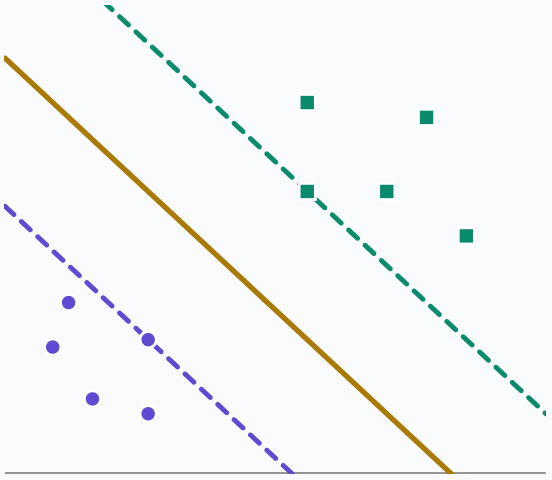
For two classes labelled $y \in \{-1, +1\}$ (note: not 0/1), a linear classifier computes a score and takes its sign:

$$f(x) = w^\top x + b, \qquad \hat{y} = \operatorname{sign}\big(f(x)\big)$$

If the classes can be separated by a line, infinitely many lines work. Which is best?



Many lines separate the training data, but some pass very close to points.



The SVM picks the line in the middle of the **widest empty street** between the classes.

WHY A WIDE MARGIN?

A line squeezed against the training points could be tipped over by a little noise in new data. A line in the middle of a wide gap tolerates more. A wider margin means a simpler, more robust classifier: this is the link to regularisation.

How SVMs fit into the course

	LOGISTIC REGRESSION	LINEAR SVM
Labels	$\{0, 1\}$	$\{-1, +1\}$
Loss	Cross-entropy	Hinge loss (6.6)
Output	A probability $P(Y = 1 \mid x)$	A score / class only (probabilities need extra calibration)
Which points matter	All points	Only support vectors: points on or inside the margin
Regularisation	λ (ridge or LASSO)	C (acts like $1/\lambda$), L2 on w
Nonlinear boundaries	Add features by hand	Kernels (6.7)
Needs standardised features	For penalties: yes	Yes

Both are linear classifiers with a boundary $w^\top x + b = 0$. They differ in which line they choose, because they use different losses.

Cheatsheet

Logistic regression vs linear SVM

Labels 0/1 vs ±1 · cross-entropy vs hinge · probability vs score · all points vs support vectors · λ vs C · features vs kernels.

SVM decision rule

$f(x) = w^\top x + b$; predict $\text{sign}(f(x))$. Labels $y \in \{-1, +1\}$.

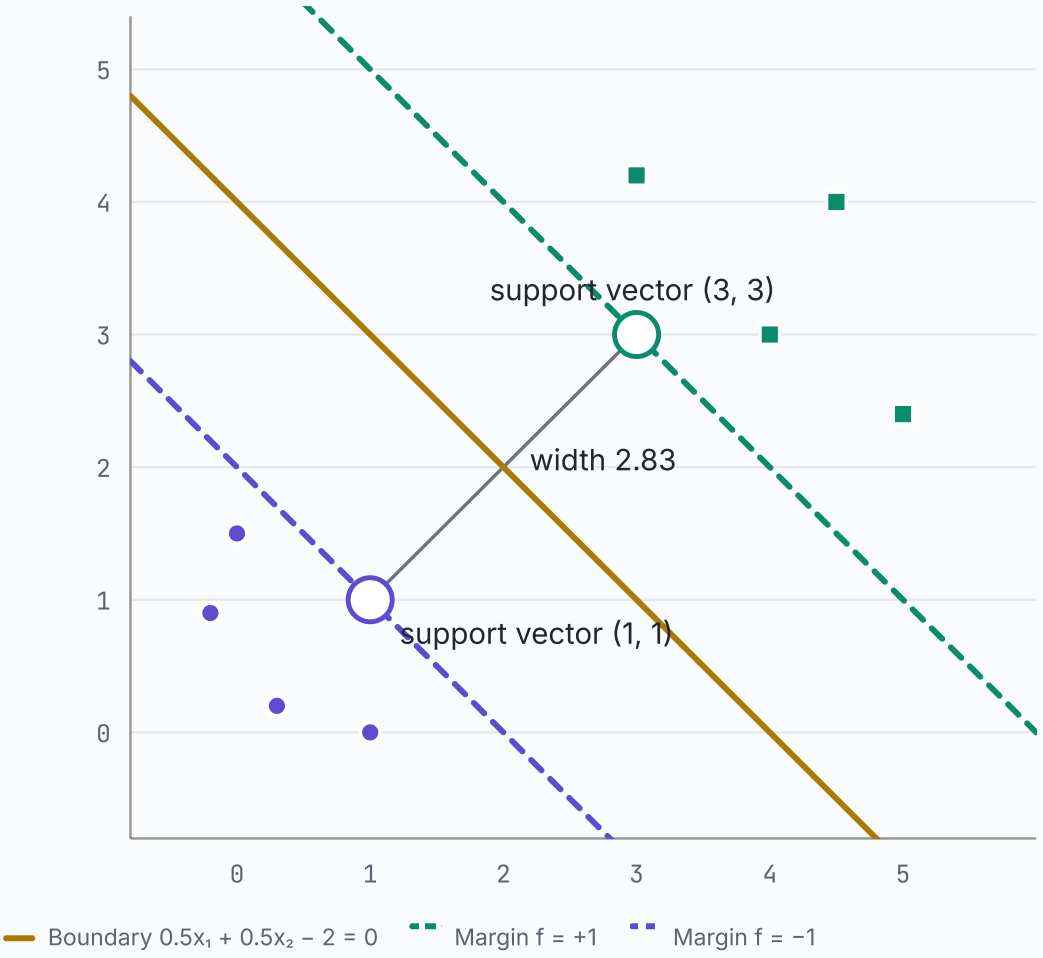
Online version with the interactive quiz: airesearchlabs.si/learn/ml/clustering-and-svm/svm-fundamentals

Margins and Support Vectors

The geometry of the margin: distance from a point to the boundary, the margin width 2/‖w‖, what support vectors are, and a complete SVM solved by hand.

Geometry

- The **decision boundary** is the hyperplane $w^\top x + b = 0$.
- The two **margin lines** are $w^\top x + b = +1$ and $w^\top x + b = -1$.
- The distance from a point x to the boundary is $\frac{|w^\top x + b|}{\|w\|}$.
- So the **margin width** (gap between the two margin lines) is $\frac{2}{\|w\|}$. A wider margin means a **smaller** $\|w\|$.



Circled points sit exactly on the margin lines: they are the support vectors. Moving any other point (without crossing the margin) leaves the solution unchanged.

Support vectors

Support vectors are the training points lying exactly on the margin lines (and, in the soft-margin case, inside the margin or misclassified). They alone determine the solution: delete or move any other point, as long as it stays outside the margin, and the boundary doesn't change. That's why SVMs can be efficient and robust to points far from the boundary.

AN SVM BY HAND

Negative support vector $(1, 1)$, positive support vector $(3, 3)$, other points further away. By symmetry, $w = (a, a)$. Support vectors satisfy $f = \pm 1$:

$$f(1,1) = 2a + b = -1, \qquad f(3,3) = 6a + b = +1$$

Subtract: $4a = 2$, so $a = 0.5$ and $b = -2$. Hence $w = (0.5, 0.5)$, $b = -2$.

$\|w\| = \sqrt{0.25 + 0.25} = 0.707$, so margin width $= 2/0.707 = 2.83$. Check: it equals the distance between the two support vectors, $\sqrt{4 + 4} = 2.83$.

Classify $(2, 3)$: $f = 0.5(2) + 0.5(3) - 2 = 0.5 > 0$, so $+1$. Its distance to the boundary is $0.5/0.707 = 0.71$. Because $|f| < 1$, it lies *inside* the margin.

MARGIN FROM A GIVEN W, B

$w = (3, 4)$, $b = -5$: $\|w\| = 5$, margin width $= 2/5 = 0.4$. Point $(2, 1)$: $f = 6 + 4 - 5 = 5 > 0$, class $+1$, distance $= 5/5 = 1$, outside the margin since $|f| \geq 1$.

Cheatsheet

SVM geometry

Boundary $w^\top x + b = 0$; margins $= \pm 1$. Distance $\frac{|w^\top x + b|}{\|w\|}$. Width $\frac{2}{\|w\|}$.

Support vectors

Points on (or inside) the margin. They alone determine the boundary.

SVM by hand (2 support vectors)

Use symmetry for the direction of w; set $f = -1$ at the negative SV and $f = +1$ at the positive SV; solve; width $= 2/\|w\|$.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/clustering-and-svm/margins-and-support-vectors

Hard-margin and Soft-margin SVM

The hard-margin optimisation problem, why it fails on overlapping data, slack variables and hinge loss, the soft-margin objective, and how hinge loss compares with logistic and 0–1 loss.

Hard margin

HARD-MARGIN SVM (PERFECTLY SEPARABLE DATA)

$$\min_{w,b} \ \tfrac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^\top x_i + b) \geq 1 \ \text{ for all } i$$

The constraint says every point is on the correct side and outside the margin. Minimising $\|w\|^2$ maximises the width $2/\|w\|$ (squaring and halving just make the maths easier).

The problem: real data are rarely separable. One overlapping point makes the constraints impossible, and a single outlier can force a very narrow margin.

Soft margin and hinge loss

Allow violations, but charge for them. For each point define the **hinge loss** (or slack):

$$\xi_i = \max\left(0,\ 1 - y_i f(x_i)\right)$$

$Y_I F(X_I)$	WHERE THE POINT IS	Ξ_I
≥ 1	Correct side, outside the margin	0
between 0 and 1	Correct side, inside the margin	between 0 and 1
< 0	Misclassified	> 1

SOFT-MARGIN SVM

$$\min_{w,b} \ \underbrace{\tfrac{1}{2}\|w\|^2}_{\text{wide margin (simplicity)}} + C \ \underbrace{\sum_{i=1}^{n} \xi_i}_{\text{hinge loss on training data}}$$

Equivalently: $\min \frac{1}{n}\sum_i \max(0, 1 - y_i f(x_i)) + \lambda\|w\|^2$ with $\lambda \propto 1/C$. That's hinge loss plus an L2 penalty, the ridge idea applied to classification.

HINGE LOSS AND THE OBJECTIVE BY HAND

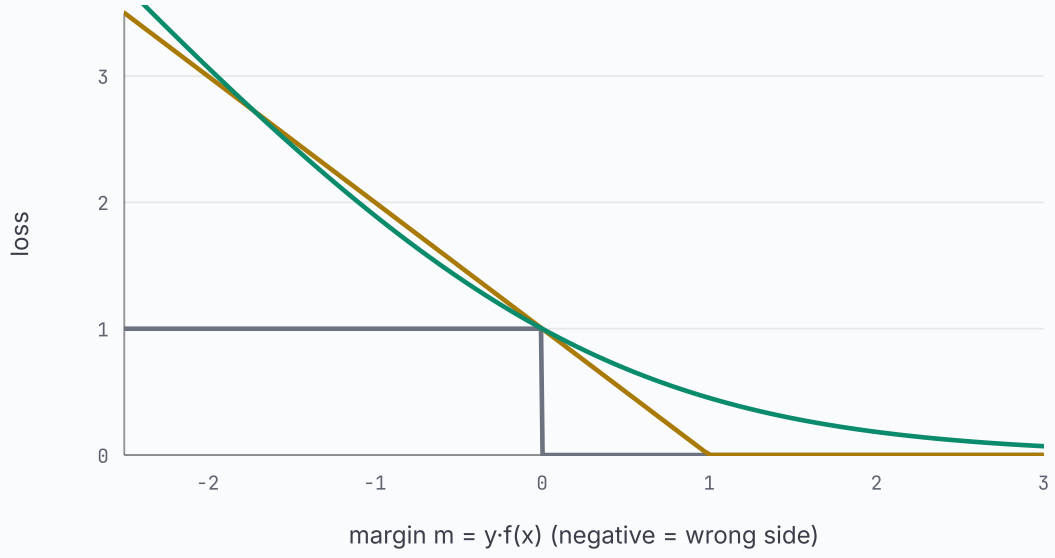
$y = +1$: $f = 2 \to 0$; $f = 0.3 \to 0.7$; $f = -0.5 \to 1.5$. $y = -1$, $f = -2$: $\max(0, 1 - 2) = 0$.

Suppose $w = (1, -1)$, $b = 0$ and four points have $\xi = (0, 0, 1, 0.5)$. Then $\frac{1}{2}\|w\|^2 = \frac{1}{2}(1 + 1) = 1$ and $\sum \xi = 1.5$.

$C = 1$: objective $= 1 + 1.5 = 2.5$. $C = 2$: objective $= 1 + 3 = 4$. A larger C makes violations count more, pushing toward a narrower margin with fewer violations.

Hinge vs the other classification losses

Write each loss in terms of the margin $m = yf(x)$ (positive = correct):



Hinge loss is exactly zero once a point is beyond the margin ($m \geq 1$); logistic loss is smooth and never exactly zero. Both are convex stand-ins for the 0–1 loss, which is hard to optimise.

Because hinge loss is zero for points comfortably on the right side, only margin violators (the support vectors) influence the SVM. Logistic regression is influenced by every point.

Cheatsheet

Hard-margin SVM

$\min \frac{1}{2}\|w\|^2$ s.t. $y_i(w^\top x_i + b) \geq 1$.

Soft-margin SVM

$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$, $\xi_i = \max(0, 1 - y_i f(x_i))$. ≡ hinge + L2, $\lambda \propto 1/C$.

Hinge loss

$\max(0, 1 - yf(x))$. 0 beyond the margin; 0–1 inside; >1 if misclassified.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/clustering-and-svm/hard-margin-and-soft-margin-svm

The C Parameter and Kernels

C trades margin width against training violations, exactly like an inverse regularisation strength. Kernels let an SVM draw curved boundaries by working in a richer feature space. Linear, polynomial and RBF kernels, and tuning C and γ.

The role of C

	LARGE C (SMALL λ)	SMALL C (LARGE λ)
Margin	Narrow	Wide
Training violations	Few tolerated	Many tolerated
Bias / variance	Low bias, high variance (overfitting risk)	High bias, low variance (underfitting risk)
Support vectors	Fewer	More
Analogue	$\lambda \to 0$ in ridge/LASSO	Large λ

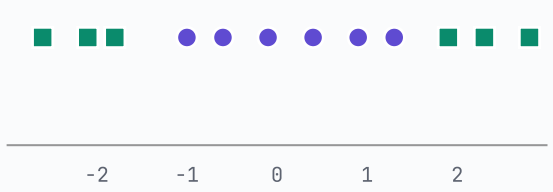
C is a hyperparameter: choose it by cross-validation, on standardised features.

DIRECTION CHECK

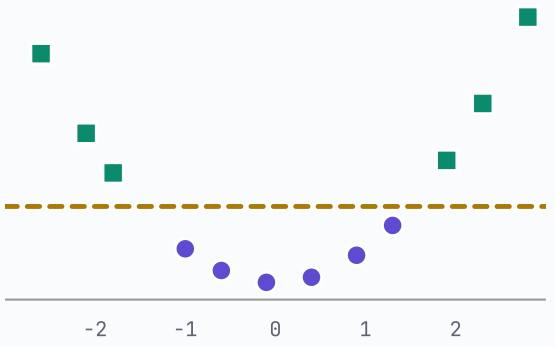
C multiplies the *training error*, while λ multiplies the *penalty*. So more C means less regularisation. It's easy to get backwards.

Kernels: curved boundaries

If no straight line separates the classes, map x into a richer feature space $\phi(x)$ where a hyperplane does.



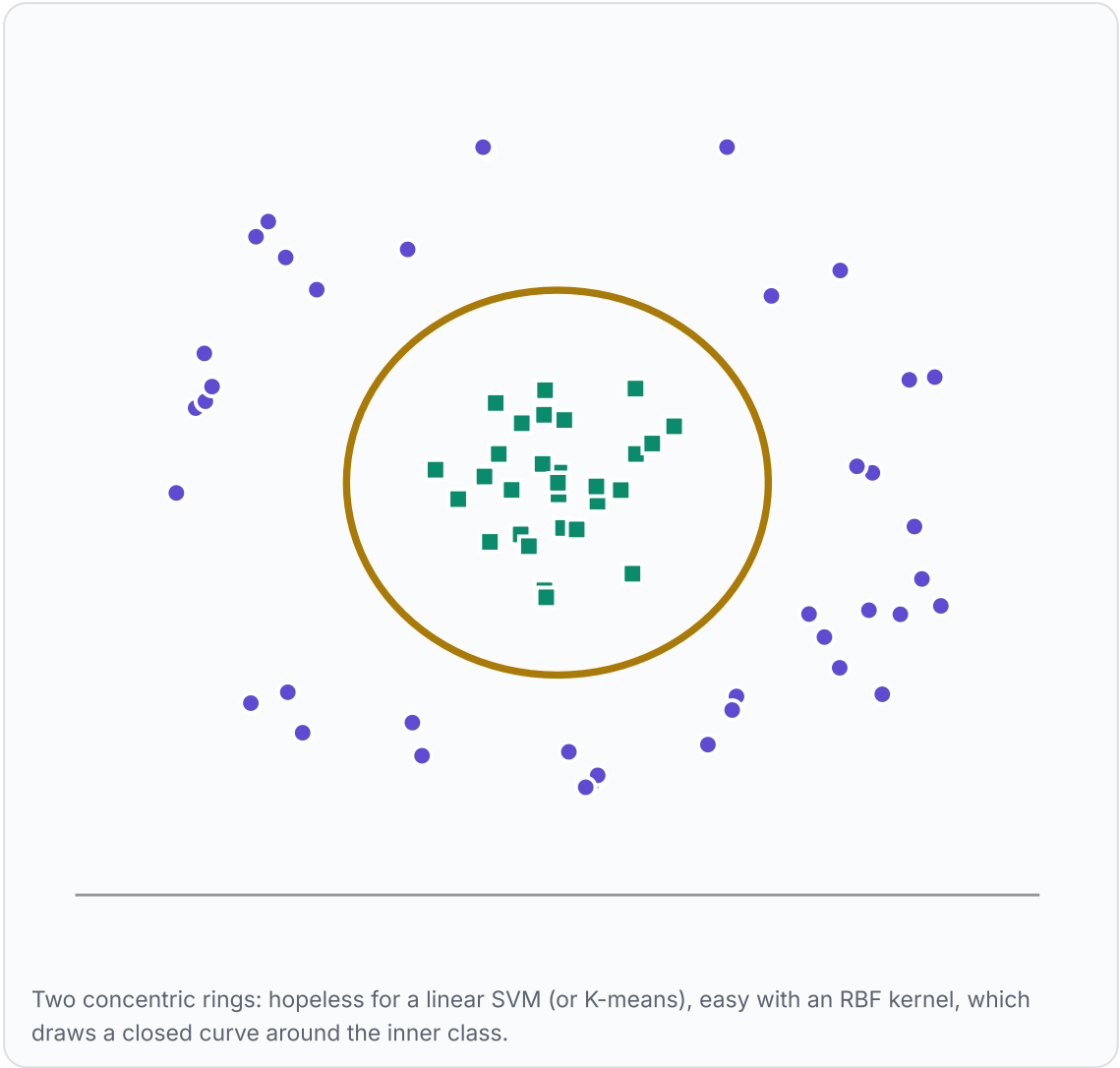
In 1-D, the outer class surrounds the inner one: no single cut-point separates them.



Map x → (x, x²). Now a straight line (x² = 2.25) separates them perfectly.

The **kernel trick** makes this cheap. The SVM's solution only ever needs inner products between points, so we can use a kernel function $K(x, x') = \phi(x)^\top \phi(x')$ that computes them directly, without ever building $\phi(x)$, which can even be infinite-dimensional.

KERNEL	$K(X, X')$	BOUNDARY
Linear	$x^\top x'$	Straight line / hyperplane
Polynomial	$(x^\top x' + c)^d$	Polynomial curves of degree d
RBF (Gaussian)	$\exp(-\gamma \lVert x - x' \rVert^2)$	Flexible, local "bumps"



Two concentric rings: hopeless for a linear SVM (or K-means), easy with an RBF kernel, which draws a closed curve around the inner class.

Tuning γ in the RBF kernel

- **Large γ:** each training point's influence is very local, so the boundary is wiggly: low bias, high variance (overfitting).
- **Small γ:** influence spreads far, so the boundary is smooth: high bias (underfitting).
- Tune C and γ **together** by cross-validation (e.g. a grid over both).

Practical notes

- Standardise features: distances and inner products are scale-sensitive.

- For class imbalance, use class weights (a bigger C for the rare class).
- Multi-class problems use one-vs-rest or one-vs-one combinations of binary SVMs.
- Training cost grows quickly with n, and probabilities need an extra calibration step.

Cheatsheet

Effect of C

Large C: narrow margin, few violations, low bias/high variance. Small C: wide margin, high bias/low variance. C ~ 1/λ.

Kernels

Linear $x^\top x'$ · Polynomial $(x^\top x' + c)^d$ · RBF $e^{-\gamma\|x-x'\|^2}$.

Complexity knobs, all models

More complex: ↑ degree, ↑ #features, ↓ λ, ↓ K (K-NN), ↑ C, ↑ γ (RBF), ↑ tree depth.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/clustering-and-svm/c-parameter-and-kernels