

Support Vector Machines (SVMs)

Aayush Bajaj

July 10, 2026

Contents

1	the maximum-margin primal	1
1.1	setup	1
1.2	normalising the scale away	2
2	the lagrangian dual, carefully	2
2.1	building the lagrangian	2
2.2	eliminating the primal variables	2
2.3	kkt conditions and support vectors	3
3	soft margins: the c-formulation	3
4	the hinge-loss view	4
5	the kernel trick	4
6	solving it: an smo sketch	5
7	in code	5
8	see also	7

a linearly separable dataset admits infinitely many separating hyperplanes, and the **perceptron** will happily hand you whichever one it trips over first. the support vector machine asks a better question: of all the hyperplanes that separate the data, which one is *farthest from everybody*? the answer — the maximum-margin hyperplane — is determined by a handful of boundary points (the support vectors), drops out of a beautiful convex dual, and generalises via the kernel trick from lines to nearly anything.

the perceptron's answer depends on initialisation and the order you feed it data. the svm's answer is unique.

1 the maximum-margin primal

1.1 setup

labels $y_i \in \{-1, +1\}$, linear score $f(x) = w^\top x + b$, classification by $\text{sign } f(x)$. the euclidean distance from a point x_i to the hyperplane $\{x : w^\top x + b = 0\}$ is

$$\frac{|w^\top x_i + b|}{\|w\|} = \frac{y_i(w^\top x_i + b)}{\|w\|} \quad \text{when correctly classified.} \quad (1)$$

the **geometric margin** of a separating hyperplane is the smallest such distance

over the training set. maximise it:

$$\max_{w,b} \min_i \frac{y_i(w^\top x_i + b)}{\|w\|}. \quad (2)$$

1.2 normalising the scale away

(w, b) and (cw, cb) describe the same hyperplane, so we may fix the scale by demanding $\min_i y_i(w^\top x_i + b) = 1$. the closest points then sit on the two *margin hyperplanes* $w^\top x + b = \pm 1$, the geometric margin is $1/\|w\|$, and maximising it is minimising the norm (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020):

$$\boxed{\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad y_i(w^\top x_i + b) \geq 1, \quad i = 1, \dots, n.} \quad (3)$$

a quadratic objective with affine constraints: a convex QP with a unique solution — no local minima, no initialisation folklore.

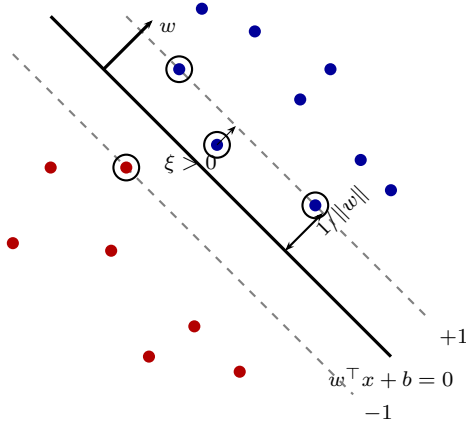


Figure 1: the margin geometry: solid decision boundary $w^\top x + b = 0$, dashed margin hyperplanes at ± 1 , support vectors circled. one point sits inside the margin with slack $\xi > 0$ — legal under the soft-margin formulation, at a price of $C\xi$.

2 the lagrangian dual, carefully

2.1 building the lagrangian

introduce a multiplier $\alpha_i \geq 0$ for each inequality constraint:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i [y_i(w^\top x_i + b) - 1]. \quad (4)$$

the primal problem is $\min_{w,b} \max_{\alpha \geq 0} \mathcal{L}$; the dual swaps the order. because the objective is convex and the constraints affine, slater's condition holds and **strong duality** applies — the swap costs nothing (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020).

2.2 eliminating the primal variables

$\mathcal{L}$ is convex in (w, b) , so minimise by stationarity:

$$\begin{aligned}\nabla_w \mathcal{L} = w - \sum_i \alpha_i y_i x_i = 0 &\implies w = \sum_i \alpha_i y_i x_i, \\ \frac{\partial \mathcal{L}}{\partial b} = -\sum_i \alpha_i y_i = 0 &\implies \sum_i \alpha_i y_i = 0.\end{aligned}$$

the optimal w is a **linear combination of training points**. substitute both back in; the b term dies by the second condition, and after collecting terms:

$$\boxed{\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^\top x_j \quad \text{s.t.} \quad \alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0.} \quad (5)$$

a QP in n variables in which the data enter **only through inner products** $x_i^\top x_j$ — remember this, it becomes the kernel trick.

2.3 kkt conditions and support vectors

at the joint optimum the karush–kuhn–tucker conditions hold (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009):

- **stationarity** — $w = \sum_i \alpha_i y_i x_i$, $\sum_i \alpha_i y_i = 0$.
- **primal feasibility** — $y_i(w^\top x_i + b) \geq 1$.
- **dual feasibility** — $\alpha_i \geq 0$.
- **complementary slackness** — $\alpha_i [y_i(w^\top x_i + b) - 1] = 0$ for every i .

the last line is the punchline. for each point, either $\alpha_i = 0$ or the constraint is *active*, $y_i(w^\top x_i + b) = 1$. so:

- points strictly outside the margin have $\alpha_i = 0$ — they contribute nothing to w . delete them and the solution does not move.
- the **support vectors** — the points on (or later, inside) the margin — carry all the α mass and alone determine the boundary.
- b is recovered from any margin support vector via $y_s(w^\top x_s + b) = 1$, i.e. $b = y_s - w^\top x_s$ (numerically: average this over all margin sv's).

compare logistic regression, where every point tugs on the solution forever.

3 soft margins: the c-formulation

real data overlap. give each point a slack $\xi_i \geq 0$ measuring how far it fails the margin, and charge for the total (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009):

$$\min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.t.} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0. \quad (6)$$

running the same lagrangian machinery (multipliers α_i for the margin constraints, μ_i for $\xi_i \geq 0$; stationarity in ξ_i gives $\alpha_i = C - \mu_i$) yields **the identical dual** with one change — the box constraint:

$$0 \leq \alpha_i \leq C. \quad (7)$$

the α value now reads as a diagnosis:

α_i	where the point sits
$\alpha_i = 0$	outside the margin, ignored
$0 < \alpha_i < C$	exactly on the margin ($\xi_i = 0$)
$\alpha_i = C$	on or inside the margin, possibly misclassified ($\xi_i \geq 0$)

C is the inverse-regularisation dial: $C \rightarrow \infty$ recovers the hard margin (violations infinitely expensive, low bias, high variance); small C buys a wide, calm margin that tolerates violations. choose it by **cross validation**.

4 the hinge-loss view

move the constraints into the objective: at the optimum $\xi_i = \max(0, 1 - y_i f(x_i))$, so the soft-margin svm is *exactly*

$$\min_{w, b} \sum_{i=1}^n \underbrace{\max(0, 1 - y_i(w^\top x_i + b))}_{\text{hinge loss}} + \frac{\lambda}{2} \|w\|^2, \quad \lambda = 1/C, \quad (8)$$

which is empirical risk minimisation with an ℓ_2 penalty — the svm rejoins the regularised-loss family alongside ridge and logistic regression (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009). the hinge is a convex upper bound on the 0–1 loss that is *exactly zero* past the margin — that flat region is why solutions are sparse in the data. the comparative anatomy of hinge vs log vs exponential loss lives on the **loss functions** page.

5 the kernel trick

the dual objective and the decision function touch data only through inner products:

$$f(x) = \sum_{i \in \text{sv}} \alpha_i y_i x_i^\top x + b. \quad (9)$$

replace every $x_i^\top x_j$ with a positive-definite kernel $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ and you have silently run the svm in the feature space of ϕ — without ever computing ϕ , which may be infinite-dimensional (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020):

- **polynomial** $k(x, z) = (1 + x^\top z)^d$: feature space of all monomials up to degree d .
- **rbf/gaussian** $k(x, z) = \exp(-\gamma \|x - z\|^2)$: infinite-dimensional feature space; γ plays bandwidth, and with it the usual bias-variance dial returns.
- the margin story survives untouched because it only ever needed a geometry of inner products.

which functions are legal kernels (mercer’s condition), what the feature spaces look like, and the representer theorem that explains why w was always going to be a combination of training points — all on the **kernel methods** page.

6 solving it: an smo sketch

the dual is a QP in n variables with a box constraint and a single linear equality. off-the-shelf QP solvers scale poorly in n ; **sequential minimal optimisation** exploits the structure.¹

- **observation** the equality $\sum_i \alpha_i y_i = 0$ means you cannot move one α alone — the smallest working set is a *pair*.
- **pair solve** fix all but (α_i, α_j) . the equality pins their weighted sum, so the subproblem is one-dimensional: a parabola in α_j over a box-induced interval $[L, H]$. the unconstrained optimum is

$$\alpha_j \leftarrow \alpha_j - \frac{y_j(E_i - E_j)}{\eta}, \quad \eta = 2k_{ij} - k_{ii} - k_{jj} \leq 0, \quad (10)$$

where $E_i = f(x_i) - y_i$ is the current error; clip to $[L, H]$, then update α_i to preserve the equality, then refresh b from the kkt conditions.

- **loop** choose pairs (heuristically: biggest kkt violators first), solve analytically, repeat until every point satisfies kkt within tolerance.
- no matrix inversions, no line searches — each step is exact and closed-form. this is what made svms practical on large datasets in the pre-gpu era.

7 in code

a “simplified smo” (random second choice, linear kernel) from scratch on two overlapping gaussian blobs:

```
1 import numpy as np
2
3 rng = np.random.default_rng(3)
4 # two gaussian blobs, slightly overlapping
5 n_half = 30
6 Xp = rng.normal([2.0, 2.0], 1.0, (n_half, 2))
7 Xm = rng.normal([-1.0, -1.0], 1.0, (n_half, 2))
8 X = np.vstack([Xp, Xm])
9 y = np.hstack([np.ones(n_half), -np.ones(n_half)])
10
11 def smo(X, y, C=1.0, tol=1e-4, max_passes=30, seed=0):
12     """platt's simplified SMO, linear kernel."""
13     rng = np.random.default_rng(seed)
14     n = len(y)
15     K = X @ X.T
16     alpha, b = np.zeros(n), 0.0
17     f = lambda i: (alpha * y) @ K[:, i] + b
18     passes = 0
19     while passes < max_passes:
20         changed = 0
21         for i in range(n):
22             Ei = f(i) - y[i]
23             if (y[i] * Ei < -tol and alpha[i] < C) or (y[i] * Ei > tol and
24                 ↪ alpha[i] > 0):
25                 j = rng.integers(n - 1)
26                 j += (j >= i) # pick j != i
27                 Ej = f(j) - y[j]
28                 ai, aj = alpha[i], alpha[j]
```

¹platt (1998), *sequential minimal optimization: a fast algorithm for training support vector machines*, microsoft research technical report msr-tr-98-14.

```

28     if y[i] != y[j]:
29         L, H = max(0, aj - ai), min(C, C + aj - ai)
30     else:
31         L, H = max(0, ai + aj - C), min(C, ai + aj)
32     if L == H:
33         continue
34     eta = 2 * K[i, j] - K[i, i] - K[j, j]
35     if eta >= 0:
36         continue
37     alpha[j] = np.clip(aj - y[j] * (Ei - Ej) / eta, L, H)
38     if abs(alpha[j] - aj) < 1e-5:
39         continue
40     alpha[i] = ai + y[i] * y[j] * (aj - alpha[j])
41     b1 = b - Ei - y[i]*(alpha[i]-ai)*K[i,i] -
42     ↪ y[j]*(alpha[j]-aj)*K[i,j]
43     b2 = b - Ej - y[i]*(alpha[i]-ai)*K[i,j] -
44     ↪ y[j]*(alpha[j]-aj)*K[j,j]
45     if 0 < alpha[i] < C: b = b1
46     elif 0 < alpha[j] < C: b = b2
47     else: b = (b1 + b2) / 2
48     changed += 1
49     passes = passes + 1 if changed == 0 else 0
50     return alpha, b
51
52 C = 1.0
53 alpha, b = smo(X, y, C=C)
54 w = (alpha * y) @ X
55 sv = alpha > 1e-6
56 margin_sv = sv & (alpha < C - 1e-6) # on the margin (0 < a < C)
57 pred = np.sign(X @ w + b)
58
59 print(f"support vectors : {sv.sum()} of {len(y)} (on margin: {margin_sv.sum()},"
60 ↪ "
61     f"inside/violating: {(alpha > C - 1e-6).sum()}")
62 print(f"w = [{w[0]:.4f}, {w[1]:.4f}], b = {b:.4f}")
63 print(f"geometric margin 1/||w|| = {1/np.linalg.norm(w):.4f}")
64 print(f"training accuracy: {(pred == y).mean():.3f}")
65 print(f"dual constraint sum(alpha_i y_i) = {alpha @ y:.2e}")
66 # kkt spot check: margin sv's should have y_i f(x_i) ~ 1
67 fx = X @ w + b
68 print("margin sv's, y_i f(x_i):", np.round((y * fx)[margin_sv], 3))

```

```

support vectors : 7 of 60 (on margin: 3, inside/violating: 4)
w = [1.4140, 1.2298], b = -1.5447
geometric margin 1/||w|| = 0.5336
training accuracy: 1.000
dual constraint sum(alpha_i y_i) = -2.00e-15
margin sv's, y_i f(x_i): [1.    1.    0.885]

```

reading the output against the theory:

- **sparsity:** 60 training points, 7 support vectors — the other 53 have $\alpha_i = 0$ exactly and could be deleted without moving the boundary.
- **the box in action:** 3 sv's sit on the margin ($0 < \alpha < C$), 4 have hit the ceiling $\alpha = C$ — the blobs overlap, so the soft margin absorbs them as paid violations.
- **the equality constraint** $\sum_i \alpha_i y_i = 0$ holds to machine precision.
- **kkt check:** margin sv's should satisfy $y_i f(x_i) = 1$; two do exactly, one reads 0.885 — the honest cost of the simplified pair-selection heuristic and loose tolerance. full smo with proper working-set selection drives every violation below 10^{-4} .

8 see also

- **the perceptron** — finds a separating hyperplane; the svm finds the best one
- **kernel methods** — the machinery that turns this linear method nonlinear
- **loss functions** — hinge loss among its rivals
- **logistic regression** — the smooth-loss sibling that keeps probabilities but loses sparsity
- **bias-variance decomposition** — what C and γ are really negotiating

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.