Support Vector Machines and Multiclass Classification

What makes a loss function good for classification?

CSCI 567 — Machine Learning • Lecture 05 • Sep 25, 2026

Logistic loss never reaches zero; this lecture starts from the margin instead

logistic regression: every point costs something, however confidently correct

Perceptron: zero loss once correct, by any margin at all

Neither says how *confidently* a point should be classified; this lecture makes confidence — the **margin** — the starting point. Today: recap of Lecture 04's kernel trick, this deck, then Lecture 04's implicit regularization and double descent.

1. Where logistic loss falls short: safely correct points keep pulling the boundary.
2. Build the **Support Vector Machine** from max-margin geometry: hard-margin, soft-margin (SVM = ERM + hinge + L2); hinge beside square and logistic as margin losses.
3. Its dual depends on the data only through inner products: Lecture 04's kernel trick, free.
4. Two classes to K: OvR, OvO, softmax (direct multiclass SVM, optional).

1. Where logistic loss falls short

The margin

Logistic loss in the ± 1 coding depends on a point only through its margin m

Labels $\mathbf{y} \in \{-1, +1\}$, score $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ (bias folded in, Lecture 02). Lecture 02's logistic loss, translated to this coding:

$$\ell_{\log}(m) \;=\; -\log \sigma(m) \;=\; \log(1 + e^{-m}),$$
$$\text{where } m \;:=\; \mathbf{y} f_{\mathbf{w}}(\mathbf{x}) \text{ is the } \textbf{functional margin}$$

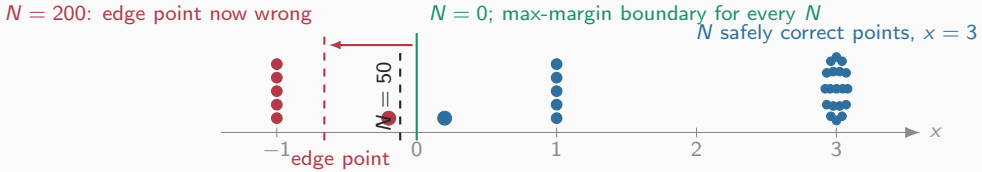
margin m	$\ell_{\log}(m)$
-1 (wrong)	1.313
0 (on the boundary)	0.693
1	0.313
3	0.049
10	4.5×10^{-5}

m is positive exactly when the point is correctly classified, and large when it is classified confidently.

strictly decreasing in m, and **never zero**: a point at $m = 10$ still costs something, and still pushes with gradient weight $\sigma(-m) \approx 4.5 \times 10^{-5}$

A safely correct cluster still moves the boundary and sacrifices an edge point

Twelve symmetric points plus N class-$+1$ points at $\mathbf{x} = 3$, all correct with a large margin from the start; ℓ_2-regularized logistic regression, $\lambda = 0.03$ (arrow: the boundary drifts as N grows):



N	boundary	margin of $\mathbf{x} = -0.2$
0	0.000	0.50
10	−0.007	0.40
50	−0.118	0.13
200	−0.660	−0.59 (wrong)

Every cluster point is already right, yet each still pays a little and pulls; 200 small pulls beat the one edge point, and the boundary moves into the other class. The max-margin boundary (midpoint of ± 0.2) stays at 0 for every N: far points do not enter it at all.

Same root cause, one more symptom: on separable data the loss falls forever

Three points, $\mathbf{x} = -1$ ($\mathbf{y} = -1$), $\mathbf{x} = 1$ ($\mathbf{y} = +1$), $\mathbf{x} = 10$ ($\mathbf{y} = +1$): separable by any $w > 0$; scale one such direction up, $w = c$:

scale c	mean logistic loss	loss of the $\mathbf{x} = 10$ point
1	0.209	4.5×10^{-5}
3	0.032	9.4×10^{-14}
30	6.2×10^{-14}	5.1×10^{-131}

no minimizer: $\|\mathbf{w}\| \to \infty$ along *any* separating direction drives the loss to 0, so the loss alone never says which separating hyperplane to prefer

Wanted: a loss that is **exactly zero** once a point is confidently correct (so it stops pulling), and a principled choice among separating hyperplanes — both from confidence measured geometrically: the **margin**.

2. Hard-margin SVM

Maximizing the margin

Two conventions we switch between: where the bias lives, how labels are coded

The bias b. Two equivalent ways to write a score:

- **explicit**: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, $\mathbf{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$.
- **folded in**: append a constant 1 to every input, $\mathbf{x} \leftarrow (\mathbf{x}, 1)$, $\mathbf{w} \leftarrow (\mathbf{w}, b)$; then $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, $\mathbf{w} \in \mathbb{R}^{d+1}$ (Lecture 02) — a column of 1s in the design matrix X.

folded in: ML theory and most lectures (one symbol, one gradient); explicit: practice and code (separate `weight`/`bias` tensors, b not penalized)

The labels $\mathbf{y}$. Two codings of the same two classes:

- $\mathbf{y} \in \{-1, +1\}$: theory, Perceptron, SVM — the margin $\mathbf{y} f(\mathbf{x})$ reads cleanly.
- $\mathbf{y} \in \{0, 1\}$: logistic regression, cross-entropy, code — $\mathbf{y}$ doubles as a probability.

linked by $\mathbf{y}_{\{0,1\}} = (\mathbf{y}_{\{\pm 1\}} + 1)/2$ (Lecture 02's bridge); neither changes the classifier

We say which convention is in force, and switch when a formula prefers it.

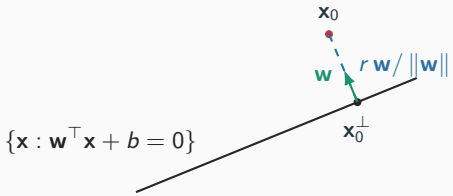
The margin is a geometric distance, so the bias b is kept separate from $\mathbf{w}$

Lectures 02–04 folded the bias into $\mathbf{w}$ via an augmented feature vector, $\mathbf{x} \leftarrow (\mathbf{x}, 1)$, $\mathbf{w} \leftarrow (\mathbf{w}, b)$ — pure notational convenience, since none of ridge, logistic regression, or kernel ridge regression's math depended on separating $\mathbf{w}$ from b.

SVM's margin is a genuine **geometric distance** in $\mathbb{R}^d$ — the distance formula only comes out right if $\|\mathbf{w}\|$ excludes the bias direction

So: $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, $\mathbf{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$, kept explicit from here on.

Point-to-hyperplane distance is $|\mathbf{w}^\top \mathbf{x}_0 + b| / \|\mathbf{w}\|$: statement and proof



schematic: the argument is coordinate-free, shown here for one orientation.

Fact. For a hyperplane $\{\mathbf{x} : \mathbf{w}^\top \mathbf{x} + b = 0\}$ and any point $\mathbf{x}_0$, the (unsigned) distance from $\mathbf{x}_0$ to the hyperplane is $|\mathbf{w}^\top \mathbf{x}_0 + b| / \|\mathbf{w}\|$

Proof. Let $\mathbf{x}_0^\perp$ be $\mathbf{x}_0$'s orthogonal projection onto the hyperplane; write $\mathbf{x}_0 = \mathbf{x}_0^\perp + r\,\mathbf{w}/\|\mathbf{w}\|$ for the signed distance r along the unit normal. Since $\mathbf{x}_0^\perp$ lies on the hyperplane, $\mathbf{w}^\top \mathbf{x}_0^\perp + b = 0$, so

$$\mathbf{w}^\top \mathbf{x}_0 + b = \mathbf{w}^\top \mathbf{x}_0^\perp + b + r\,\|\mathbf{w}\| = r\,\|\mathbf{w}\|\,,$$

so $r = (\mathbf{w}^\top \mathbf{x}_0 + b)/\|\mathbf{w}\|$ and the distance is $|r|$. ■

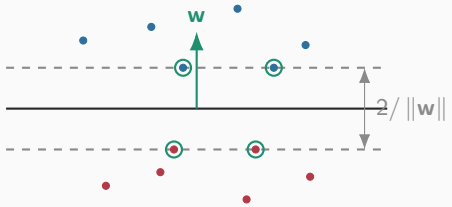
Geometric margin of a point, and of the hyperplane (the minimum over points)

For a labeled point $(\mathbf{x}_i, \mathbf{y}_i)$ correctly classified by the hyperplane, multiply the signed distance by $\mathbf{y}_i$ — turns it into a positive number:

$$\gamma_i \;=\; \frac{\mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Exactly Lecture 02's Novikoff-theorem γ, generalized: there, $\gamma_i = \mathbf{y}_i \langle \mathbf{u}, \mathbf{x}_i \rangle$ for a unit vector $\mathbf{u}$; here $\mathbf{w}/\|\mathbf{w}\|$ is that same unit vector, made explicit.

The hyperplane's margin: $\gamma = \min_i \gamma_i$, its distance to the closest point.



solid: hyperplane; dashed: margin boundaries; blue: $\mathbf{y} = +1$, red: $\mathbf{y} = -1$; circled: points exactly on the margin (named **support vectors** in Section 4).

The max-margin problem, and step 1: scale so the closest point has margin 1

Among all hyperplanes that separate the two classes, hard-margin SVM picks the one with the largest margin — not yet a form calculus can attack, since $\|\mathbf{w}\|$ sits in every denominator:

$$\max_{\mathbf{w},b} \ \gamma \qquad \text{s.t.} \qquad \frac{\mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b)}{\|\mathbf{w}\|} \ \geq \ \gamma \ \ \forall i$$

Step 1: scale invariance. $(\mathbf{w}, b)$ and $(c\mathbf{w}, cb)$ for any $c > 0$ describe the identical hyperplane and the identical γ — numerator and denominator of every γ_i both scale by c.

so pick the scaling with $\min_i \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$ — always achievable by dividing by that minimum (positive, since every point is correctly classified)

Steps 2–3: now $\gamma = 1/\|\mathbf{w}\|$, so maximizing γ means minimizing $\|\mathbf{w}\|$

Step 2. The constraint set becomes $\mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$ for every i, with equality for the closest point, so

$$\gamma \quad = \quad \min_i \frac{\mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b)}{\|\mathbf{w}\|} \quad = \quad \frac{1}{\|\mathbf{w}\|}$$

since the numerator's minimum is exactly 1 by Step 1's construction.

Step 3.

$$\max \frac{1}{\|\mathbf{w}\|} \quad \Leftrightarrow \quad \min \|\mathbf{w}\| \quad \Leftrightarrow \quad \min \tfrac{1}{2} \|\mathbf{w}\|_2^2$$

Squaring and halving don't change the minimizer — purely for clean derivatives, Lecture 02's $\frac{1}{2}$-in-the-MSE trick.

Standard-form hard-margin SVM: a convex QP with linear constraints

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|_2^2 \qquad \text{s.t.} \qquad \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \ \ \forall i$$

Cortes and Vapnik (1995).

a convex quadratic objective with linear constraints
— uniquely solvable, given a linearly separable dataset

Discussion 05 works a small instance of this by hand.

3. Soft-margin SVM

$$\text{SVM} = \text{ERM} + \text{hinge} + \text{L2}$$

Hard-margin SVM is fragile; slack variables $\xi_i \geq 0$ relax each constraint

- **Not separable**: no solution at all — a single misclassified point makes every constraint infeasible simultaneously.
- **Separable, but barely**: hard-margin SVM can pick a needlessly narrow margin to accommodate one nearby outlier.

Section 1's problem in mirror image: there separable data broke logistic loss, here non-separable data breaks the constraints

One slack variable $\xi_i \geq 0$ per training point:

$$\mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

$\xi_i = 0$: at or beyond the margin, as before

$0 < \xi_i \leq 1$: inside the margin, still correctly classified

$\xi_i > 1$: misclassified

Soft-margin SVM pays C per unit of slack; the optimal slack is the hinge

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i \qquad \text{s.t.} \qquad \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0 \;\; \forall i$$

$C > 0$: large C penalizes slack heavily (pushes toward hard-margin where feasible); small C tolerates more violation for a wider margin.

For any fixed $(\mathbf{w}, b)$, the objective is smallest when each ξ_i is as small as possible subject to feasibility:

$$\xi_i^\star \;=\; \max\big(0,\; 1 \;-\; \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i \;+\; b)\big)$$

the smallest non-negative slack meeting the relaxed constraint — the hinge loss.

Substituting the optimal slack removes the constraints: the ERM + L2 template

Plugging in $\xi_i^\star$ eliminates the constraints entirely:

$$\min_{\mathbf{w},b} \; \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\left[1 - \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b)\right]_+$$

Divide by n and set $\lambda = 1/(Cn)$:

$$\min_{\mathbf{w},b} \; \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left[1 - \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b)\right]_+}_{=\,\hat{\mathcal{R}}_{\text{hinge}}(\mathbf{w},b)} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

SVM = ERM + hinge + L2: one family with ridge and logistic regression

SVM = ERM + hinge loss + L2 regularization.

Exactly Lecture 03's ridge ($\hat{\mathcal{R}}_{\text{sq}} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$) and Lecture 02–03's ℓ_2-regularized logistic regression ($\hat{\mathcal{R}}_{\log} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$), with the loss swapped for hinge: pick a loss, optionally add an ℓ_2 penalty, minimize.

At the level of algorithms. (Stochastic sub-)gradient descent on $\hat{\mathcal{R}}_{\text{hinge}}$ alone ($\lambda = 0$) is close to the Perceptron's mistake-driven update.

Perceptron updates on mistakes ($m < 0$); SVM's hinge term also updates on correct points still inside the margin ($0 \leq m < 1$)

ℓ_2 regularization is the piece Perceptron never had — it turns "some separating hyperplane" into "the widest-margin one, softly enforced."

Square, logistic, and hinge, all written as functions of the margin m

Square (Lecture 02). With $\mathbf{y} \in \{-1, +1\}$ so $\mathbf{y}^2 = 1$:

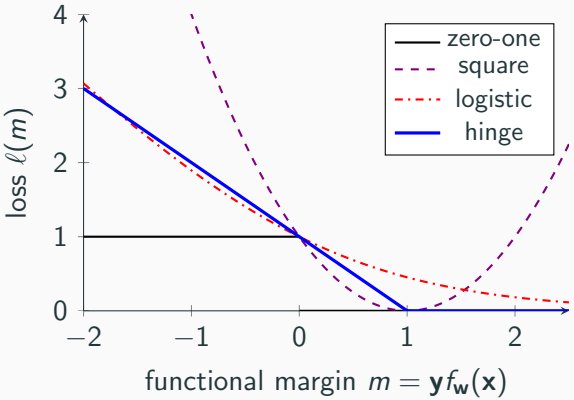
$$(f_{\mathbf{w}}(\mathbf{x}) - \mathbf{y})^2 = f_{\mathbf{w}}(\mathbf{x})^2 - 2\mathbf{y} f_{\mathbf{w}}(\mathbf{x}) + \mathbf{y}^2 = \mathbf{y}^2 f_{\mathbf{w}}(\mathbf{x})^2 - 2m + 1 = m^2 - 2m + 1 \quad \Rightarrow \quad \ell_{\text{sq}}(m) = (1-m)^2$$

Squared error *is* a margin loss, just not usually written that way — and it penalizes $m \neq 1$ symmetrically: a point at $m = 10$ costs 81, more than a misclassified point at $m = -8$.

Logistic (Lecture 02): $\ell_{\log}(m) = \log(1 + e^{-m}) / \log 2$ — always incurs *some* penalty, however large m is, never reaches 0

Hinge (this lecture): $\ell_{\text{hinge}}(m) = [1 - m]_+ := \max(0, 1 - m)$ — zero once $m \geq 1$, confidently correct *beyond* a margin of 1

Four losses side by side: all agree at $m = 0$; only hinge reaches exactly 0



zero-one $= \mathbb{1}[m \leq 0]$ is the error rate itself; the other three are convex surrogates of it, all equal to 1 at $m = 0$.

- **Square** is the only one that grows again for $m > 1$ — penalizing points already more than correctly classified.
- **Logistic and hinge** both decrease monotonically as m grows, matching 0-1 loss's indifference to overshooting.
- **Hinge** is the only one that reaches *exactly* 0, at $m = 1$, rather than only approaching it.

Hinge = Perceptron loss shifted to demand a margin; SVM asks for confidence

Lecture 02's **Perceptron loss**: $\ell_{\text{perc}}(m) = \max(0, -m) = [-m]_+$ — zero for any $m > 0$, correctly classified at any margin at all.

$$\ell_{\text{hinge}}(m) = [1 - m]_+ = \ell_{\text{perc}}(m - 1)$$

Hinge is the Perceptron loss, shifted to demand a margin of at least 1, not just a correct sign.

SVM is what you get by taking the Perceptron's mistake-driven idea and asking not just for correctness, but for **confidence**.

Solving it: the objective is convex, so subgradient descent converges

$\hat{\mathcal{R}}_{\text{hinge}}(\mathbf{w}, b) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$ is convex — hinge is a max of two linear functions, hence convex; convex + convex penalty is convex.

Lecture 02's gradient descent applies, using a **subgradient** at hinge's non-differentiable kink ($m = 1$): $-\mathbf{y}_i \mathbf{x}_i$ (and $-\mathbf{y}_i$ for b) when $m_i < 1$, else 0.

nothing new is needed to fit an SVM: Lecture 02's (stochastic) gradient descent loop, with the hinge subgradient in place of the gradient — Discussion 05 runs it on a small instance

4. SVM duality

Kernels, for free

Lecture 04's playbook, and the identity $[z]_+ = \max_{\alpha \in [0,1]} \alpha z$ that starts it

Lecture 04's kernel methods showed a problem's dual can depend on the data only through inner products.

does the same happen for soft-margin SVM's $\hat{\mathcal{R}}_{\text{hinge}} + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$?

$$[z]_+ = \max_{\alpha \in [0,1]} \alpha z$$

rewrites hinge's max in terms of a new variable α per training point.

$z \geq 0$: the max is at $\alpha = 1$, giving $z = [z]_+$

$z < 0$: the max is at $\alpha = 0$, giving $0 = [z]_+$

Immediate — both cases checked, not just claimed.

The identity gives a min-max; convex-concave structure lets min and max swap

With $m_i = \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b)$, apply the identity to each hinge term, $z_i = 1 - m_i$:

$$\min_{\mathbf{w},b} \max_{\alpha \in [0,1]^n} \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \alpha_i (1 - \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i + b))}_{=:\, G(\mathbf{w},b,\alpha)}$$

G is convex in $(\mathbf{w}, b)$ for fixed α (convex quadratic plus linear), and concave — in fact linear — in α for fixed $\mathbf{w}, b$, over the compact convex domain $[0,1]^n$.

this convex-concave, compact-domain structure licenses swapping the order of min and max (a minimax/strong-duality argument, stated here rather than proved in full): $\min_{\mathbf{w},b} \max_\alpha G = \max_\alpha \min_{\mathbf{w},b} G$

Inner minimization: over w, a weighted sum of points; over b, a constraint

Over w. Set the gradient to zero:

$$\nabla_{\mathbf{w}} G = \lambda \mathbf{w} - \frac{1}{n} \sum_i \alpha_i \mathbf{y}_i \mathbf{x}_i = 0 \qquad \Rightarrow \qquad \mathbf{w}^\star = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i \mathbf{y}_i \mathbf{x}_i$$

Over b.

$$\frac{\partial G}{\partial b} = -\frac{1}{n} \sum_i \alpha_i \mathbf{y}_i = 0 \qquad \Rightarrow \qquad \sum_{i=1}^{n} \alpha_i \mathbf{y}_i = 0$$

If this constraint fails, the inner minimum over b is $-\infty$, so the outer maximization over α never selects such an α — it becomes an equality constraint on the dual problem.

Substituting back gives the SVM dual: a concave QP in n variables, not d

Substitute $\mathbf{w}^\star$ back and simplify — the two quadratic-in-α terms combine, the b term vanishes using $\sum_i \alpha_i \mathbf{y}_i = 0$:

$$\max_{\alpha \in [0,1]^n,\ \sum_i \alpha_i \mathbf{y}_i = 0} \quad \frac{1}{n} \mathbf{1}^\top \alpha - \frac{1}{2\lambda n^2} \alpha^\top Y K Y \alpha$$

$$Y \quad = \quad \mathrm{diag}(\mathbf{y}), \quad K_{ij} \quad = \quad \mathbf{x}_i^\top \mathbf{x}_j$$

a concave quadratic program in $\alpha \in \mathbb{R}^n$ —
n variables, regardless of the input dimension d

Dual and prediction rule touch the data only via inner products

Look at what K is: exactly Lecture 04's kernel/Gram matrix,

$K_{ij} = \mathbf{x}_i^\top \mathbf{x}_j = \kappa_{\text{linear}}(\mathbf{x}_i, \mathbf{x}_j)$ — an inner product; nothing else about the raw data appears in the dual

The prediction rule inherits this. For a new point $\mathbf{x}$:

$$f_{\mathbf{w}^\star, b^\star}(\mathbf{x}) = \mathbf{w}^{\star\top}\mathbf{x} + b^\star = \frac{1}{\lambda n}\sum_{i=1}^{n} \alpha_i^\star \mathbf{y}_i \, \mathbf{x}_i^\top \mathbf{x} + b^\star$$

also only ever needs inner products $\mathbf{x}_i^\top \mathbf{x}$, never $\mathbf{x}_i$ or $\mathbf{x}$ individually.

this is precisely the structural fact Lecture 04's representer theorem and kernel trick were built for

Kernel SVM: replace $\mathbf{x}_i^\top \mathbf{x}_j$ with any valid kernel, no new theory needed

Replace $\mathbf{x}_i^\top \mathbf{x}_j$ with any valid kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ — Lecture 04's polynomial or RBF kernels, or any Mercer-valid kernel built from them.

same QP, same dual variables α, same optimality structure — now fitting a nonlinear decision boundary, while every computation still only ever touches $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$

At the saddle point each α_i solves a one-variable problem: three cases by margin

At the saddle point $(\mathbf{w}^\star, b^\star, \alpha^\star)$, strong duality means $\alpha^\star$ maximizes $G(\mathbf{w}^\star, b^\star, \alpha)$ — with $\mathbf{w}^\star, b^\star$ already fixed. With them fixed, G is separable across i: each α_i individually solves

$$\max_{\alpha_i \in [0,1]} \alpha_i(1 - m_i^\star)$$

$m_i^\star > 1$ (outside the margin): $1 - m_i^\star < 0$, maximizer is $\alpha_i^\star = 0$

$m_i^\star = 1$ (exactly on the margin): $\alpha_i^\star(1 - m_i^\star) = 0$ regardless — $\alpha_i^\star$ can be anywhere in $[0, 1]$

$m_i^\star < 1$ (inside the margin, or misclassified): $1 - m_i^\star > 0$, maximizer is $\alpha_i^\star = 1$

Support vectors are the points with $\alpha_i^\star > 0$; one on the margin recovers $b^\star$

Since $\mathbf{w}^\star = \frac{1}{\lambda n} \sum_i \alpha_i^\star \mathbf{y}_i \mathbf{x}_i$, only points with $\alpha_i^\star > 0$ contribute to $\mathbf{w}^\star$ at all.

support vectors: points on or inside the margin — every other training point could be deleted without changing the solution in the slightest

$b^\star$ is not directly returned by the dual — recover it from any margin support vector (the $m_i^\star = 1$ case):

$$b^\star = \mathbf{y}_i - \sum_j \alpha_j^\star \mathbf{y}_j \, \kappa(\mathbf{x}_j, \mathbf{x}_i)/(\lambda n) \qquad \text{for any } i \text{ with } \alpha_i^\star \in (0, 1)$$

Non-support points contribute exactly zero; kernel SVM is cheap at test time

at the solution, the hinge subgradient of every point with $m_i^\star > 1$ is exactly 0, so the subgradient computed over the support vectors alone equals the subgradient over the full dataset

A real sparsity property, not just a naming convention: delete every non-support point and the optimality conditions do not change.

$$f(\mathbf{x}) = \sum_i \alpha_i^\star \mathbf{y}_i \kappa(\mathbf{x}_i, \mathbf{x})/(\lambda n) + b^\star$$

Sums only over support vectors — typically a small fraction of n, not all of it.

5. Multiclass classification

From two classes to K

One-vs-rest: K binary classifiers, predict by largest score; competitive in practice

Everything so far handles two classes; real problems often have $K > 2$ (digits, species). How does any of it — SVM, logistic regression — extend to K?

One-vs-rest (OvR). Train K independent binary classifiers $f_1, \ldots, f_K$, where f_k separates class k (relabeled $+1$) from every other class pooled together (relabeled -1); predict

$$\hat{\mathbf{y}} \;=\; \arg\max_{k \in \{1,\ldots,K\}} f_k(\mathbf{x})$$

Using the raw score, not its sign, keeps this well-defined when several classifiers say "$+1$". Cost: K binary problems, each on the full n-point dataset.

✔ Rifkin and Klautau (2004): OvR is competitive with more sophisticated multiclass methods in practice, despite class k's classifier seeing a growing label imbalance and scores not calibrated across classifiers

One-vs-one: $\binom{K}{2}$ pairwise classifiers and a majority vote; each sees less data

Train $\binom{K}{2}$ binary classifiers, one per pair of classes (i, j), each trained only on points belonging to class i or class j.

predict by **majority vote**: each pairwise classifier casts one vote, the class with the most votes wins

$O(K^2)$ classifiers rather than OvR's $O(K)$ — but each trained on a much smaller subset, only two classes' worth of data

If a binary method's training cost grows worse than linearly in sample size (true of many SVM solvers), many small problems can be cheaper in total than K large ones.

Softmax: one joint model, with cross-entropy loss by the same MLE argument

OvR and OvO both *decompose* multiclass into many binary subproblems. Softmax generalizes logistic regression directly — one weight vector $\mathbf{w}_k \in \mathbb{R}^d$ per class:

$$\mathbb{P}(\mathbf{y} = k \mid \mathbf{x}) = \frac{e^{\mathbf{w}_k^\top \mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}}}$$

Loss. Direct extension of Lecture 02's derivation: there, $\mathbf{y} \mid \mathbf{x} \sim \text{Bernoulli}(\sigma(\mathbf{w}^\top \mathbf{x}))$; here, $\mathbf{y} \mid \mathbf{x} \sim \text{Categorical}(\text{softmax}(\cdot))$. Same MLE-to-ERM argument — negative log-likelihood, averaged over the sample:

$$\ell(\{\mathbf{w}_k\}, \mathbf{x}, \mathbf{y}) = -\log \mathbb{P}(\mathbf{y} \mid \mathbf{x}) = -\mathbf{w}_{\mathbf{y}}^\top \mathbf{x} + \log \sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}}$$

Convex in $\{\mathbf{w}_k\}$, no closed form — fit by gradient descent, like every other loss in this course.

At $K = 2$, softmax with $2{\times}d$ weights $[-\mathbf{w};\ \mathbf{w}]$ is logistic regression with $\mathbf{y} \in \{0, 1\}$

Two classes coded $\mathbf{y} \in \{0, 1\}$; stack the weight vectors as a $2 \times d$ matrix with rows $\mathbf{w}_0 = -\mathbf{w}$, $\mathbf{w}_1 = +\mathbf{w}$:

$$\mathbb{P}(\mathbf{y} = 1 \mid \mathbf{x}) = \frac{e^{\mathbf{w}^\top \mathbf{x}}}{e^{-\mathbf{w}^\top \mathbf{x}} + e^{\mathbf{w}^\top \mathbf{x}}} = \frac{1}{1 + e^{-2\mathbf{w}^\top \mathbf{x}}} = \sigma(2\mathbf{w}^\top \mathbf{x})$$

Lecture 02's sigmoid model with weight $2\mathbf{w}$ (rows $\mp\mathbf{w}/2$ give $\sigma(\mathbf{w}^\top \mathbf{x})$ exactly); cross-entropy with $\mathbf{y} \in \{0, 1\}$ becomes exactly Lecture 02's logistic loss:

$$-\log \mathbb{P}(\mathbf{y} \mid \mathbf{x}) = -\big[\mathbf{y} \log \sigma(2\mathbf{w}^\top \mathbf{x}) + (1 - \mathbf{y}) \log(1 - \sigma(2\mathbf{w}^\top \mathbf{x}))\big]$$

Why one w was enough. Softmax only sees differences: adding the same vector to every $\mathbf{w}_k$ leaves $e^{\mathbf{w}_k^\top \mathbf{x}} / \sum_j e^{\mathbf{w}_j^\top \mathbf{x}}$ unchanged, so K weight vectors carry one redundant direction; fixing $\mathbf{w}_0 = -\mathbf{w}_1$ (or $\mathbf{w}_0 = \mathbf{0}$) removes it, and at $K = 2$ a single $\mathbf{w} \in \mathbb{R}^d$ is left.

Direct multiclass SVM: Crammer and Singer (2001) (optional)

SVM has a joint, non-decomposed formulation too: a single optimization with a genuinely multiclass margin notion.

require the correct class to outscore every other class by a margin:

$$\mathbf{w}_{\mathbf{y}_i}^\top \mathbf{x}_i - \mathbf{w}_k^\top \mathbf{x}_i \geq 1 - \xi_i \qquad \forall k \neq \mathbf{y}_i$$

In practice, OvR remains the more common default — substantially cheaper to train for accuracy that is typically comparable.

This frame is **optional**: the rest of the course uses OvR, OvO, and softmax; Crammer–Singer appears only as a row in the comparison table for completeness.

Four multiclass methods, side by side

Method	# classifiers	Prediction rule
One-vs-rest	K	$\arg\max_k f_k(\mathbf{x})$
One-vs-one	$\binom{K}{2}$	majority vote
Softmax	1 joint model	$\arg\max_k \mathbf{w}_k^\top \mathbf{x}$
Crammer–Singer (optional)	1 joint model	$\arg\max_k \mathbf{w}_k^\top \mathbf{x}$

OvR: cheapest, scores not calibrated. OvO: more classifiers, each on less data. Softmax: cross-entropy, not margin-based. Crammer–Singer: margin-based, one QP for all K.

6. Wrap-up

One thread, start to finish

The thread: from 0-1 loss to kernel SVM to K classes

1. Logistic loss never reaches zero: a safely correct cluster still moves the boundary, and separable data never converges $\Rightarrow$ measure confidence geometrically: the margin.
2. Max-margin geometry $\Rightarrow$ hard-margin SVM $\Rightarrow$ with slack, soft-margin SVM $=$ this course's ERM $+$ L2 template, hinge loss standing in.
3. Hinge $=$ Perceptron loss demanding confidence; square, logistic, hinge are one family of margin losses, compared side by side.
4. Duality: depends on the data only through inner products $\Rightarrow$ kernel SVM, free.
5. Multiclass: decompose (OvR, OvO) or build one joint model (softmax, Crammer–Singer).

Coming up

HW2 (Lectures 03–04) due Mon 9/28, 11:59pm.

This lecture's Colab notebook (soft-margin SVM by subgradient descent, one-vs-rest multi-class) will be released next week, after HW2 is in.

Lecture 06: Unsupervised Learning — PCA, clustering, K-means, GMM: no labels $\mathbf{y}$ at all, a genuinely different setting from every model this course has built so far

References

- Cortes, Vapnik. *Support-Vector Networks.* Machine Learning, 20(3), 1995.
- Crammer, Singer. *On the Algorithmic Implementation of Multiclass Kernel-Based Vector Machines.* JMLR, 2(Dec), 2001.
- Rifkin, Klautau. *In Defense of One-vs-All Classification.* JMLR, 5, 2004.

Classification framing and SVM content (Sections 1–4) adapted from EPFL's `ML_course`, used with permission.

Full citation details: `references/sources/epfl-l05-svm-multiclass.md`, `references/sources/lecture05-svm-multiclass-citations.md`