

Support Vector Machines, Advanced Topics

Lecture Notes Consolidated with Claude AI — CS6140 Machine Learning

Optional: additional derivations and topics for the **SVM note** (“Support Vector Machines with Kernels”). Nothing here is needed for HW3’s required problems; read it for depth, or when an optional HW3 problem points here. It uses the SVM note’s notation ($\mathbf{x}$ a $1 \times D$ row vector, $\mathbf{w}$ $D \times 1$, $y \in \{-1, +1\}$, N training points) and refers back to its sections and equations as “SVM note, §N” and “SVM note, Eq. N”.

Based on Andrew Ng’s CS229 lecture notes (Stanford; Part V, *Support Vector Machines*), Bingyu Wang’s SVM notes for this course, Platt’s SMO paper (1998), and Shalev-Shwartz et al. (Pegasos, 2007).

1 Lagrange Duality

General problem: $\min_{\mathbf{w}} F(\mathbf{w})$ subject to $g_i(\mathbf{w}) \leq 0$ ($i = 1..k$) and $h_l(\mathbf{w}) = 0$ ($l = 1..m$). Lagrangian: $L(\mathbf{w}, \alpha, \beta) = F(\mathbf{w}) + \sum_i \alpha_i g_i(\mathbf{w}) + \sum_l \beta_l h_l(\mathbf{w})$, with $\alpha_i \geq 0$.

- **Primal:** $\theta_P(\mathbf{w}) = \max_{\alpha \geq 0, \beta} L(\mathbf{w}, \alpha, \beta)$ equals $F(\mathbf{w})$ if $\mathbf{w}$ satisfies all constraints, and $+\infty$ otherwise (a violated constraint lets α_i or β_l grow without bound). So $p^* = \min_{\mathbf{w}} \theta_P(\mathbf{w}) = \min_{\mathbf{w}} \max_{\alpha \geq 0, \beta} L$ is the original problem.
- **Dual:** $\theta_D(\alpha, \beta) = \min_{\mathbf{w}} L(\mathbf{w}, \alpha, \beta)$, and $d^* = \max_{\alpha \geq 0, \beta} \theta_D = \max_{\alpha \geq 0, \beta} \min_{\mathbf{w}} L$: the same expression with min and max swapped.
- **Weak duality:** $d^* \leq p^*$ always ($\max \min \leq \min \max$). For any feasible $\mathbf{w}$ and $\alpha \geq 0$, $\theta_D(\alpha, \beta) \leq L(\mathbf{w}, \alpha, \beta) \leq F(\mathbf{w})$: every dual value is a lower bound on the primal.
- **Strong duality:** $d^* = p^*$ when F and the g_i are convex, the h_l affine, and the constraints strictly satisfiable (Slater’s condition). The SVM qualifies. The optimum is then a **saddle point** of L : a minimum in $\mathbf{w}$ and a maximum in (α, β) .

A one-dimensional example. Minimize $F(w) = (w - 7)^2$ subject to $h(w) = 2w - 8 = 0$. The only feasible point is $w = 4$, so $p^* = (4 - 7)^2 = 9$. The Lagrangian is $L(w, \beta) = (w - 7)^2 + \beta(2w - 8)$.

- *Dual function:* $\partial L / \partial w = 2(w - 7) + 2\beta = 0$ gives $w = 7 - \beta$, and substituting, $\theta_D(\beta) = \beta^2 + \beta(6 - 2\beta) = -\beta^2 + 6\beta$.
- *Weak duality:* every value of θ_D is at most 9; for example $\theta_D(0) = 0$, the unconstrained minimum of F , ignoring the constraint.
- *Strong duality:* θ_D is maximized at $\beta = 3$, with $d^* = \theta_D(3) = 9 = p^*$, and the minimizing $w = 7 - \beta = 4$ is exactly the primal solution. $(w, \beta) = (4, 3)$ is the saddle point of L .

2 The KKT Conditions and the Three Types of Points

Under strong duality, $(\mathbf{w}^*, \alpha^*, \beta^*)$ is optimal if and only if the **Karush–Kuhn–Tucker** conditions hold:

1. stationarity: $\nabla_{\mathbf{w}} L(\mathbf{w}^*, \alpha^*, \beta^*) = 0$;
2. primal feasibility: $g_i(\mathbf{w}^*) \leq 0$, $h_l(\mathbf{w}^*) = 0$;
3. dual feasibility: $\alpha_i^* \geq 0$;
4. complementary slackness: $\alpha_i^* g_i(\mathbf{w}^*) = 0$ for every i .

Soft-margin SVM. The constraints are $1 - \xi_i - y_i f(\mathbf{x}_i) \leq 0$ (multiplier α_i) and $-\xi_i \leq 0$ (multiplier r_i). Stationarity in ξ_i gives $C - \alpha_i - r_i = 0$ (§3). Complementary slackness gives $\alpha_i [y_i f(\mathbf{x}_i) - 1 + \xi_i] = 0$ and $r_i \xi_i = 0$. Then:

- $\alpha_i = 0 \Rightarrow r_i = C > 0 \Rightarrow \xi_i = 0$, and feasibility gives $y_i f(\mathbf{x}_i) \geq 1$.
- $0 < \alpha_i < C \Rightarrow r_i > 0 \Rightarrow \xi_i = 0$, and $\alpha_i > 0$ forces $y_i f(\mathbf{x}_i) = 1 - \xi_i = 1$.
- $\alpha_i = C \Rightarrow r_i = 0$ (ξ_i may be positive), and $y_i f(\mathbf{x}_i) = 1 - \xi_i \leq 1$.

These are the three rows of the table in SVM note, §3; SMO's "violation" test checks exactly them.

3 Deriving the Soft-Margin Dual

Lagrangian of (SVM note, Eq. 11), with $\alpha_i \geq 0$, $r_i \geq 0$:

$$L = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i - \sum_i \alpha_i [y_i (\mathbf{x}_i \mathbf{w} + b) - 1 + \xi_i] - \sum_i r_i \xi_i. \quad (1)$$

Derivatives: $\partial L / \partial \mathbf{w} = 0 \Rightarrow \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i^T$; $\partial L / \partial b = 0 \Rightarrow \sum_i \alpha_i y_i = 0$; $\partial L / \partial \xi_i = C - \alpha_i - r_i = 0$ for each i . Substituting, the ξ_i terms cancel exactly ($C\xi_i - \alpha_i \xi_i - r_i \xi_i = 0$), leaving the hard-margin dual objective. The only trace of the slacks is the constraint: $r_i = C - \alpha_i \geq 0$ means $\alpha_i \leq C$.

Remark. Some texts (and HW3's optional Problem 15) write the penalty as $\frac{C}{N} \sum_i \xi_i$ (an average instead of a sum). Then the bound becomes $0 \leq \alpha_i \leq C/N$; nothing else changes.

4 Deriving the SMO Update

Rename the pair α_1, α_2 and let $s = y_1 y_2$, $v_k = \sum_{l \geq 3} \alpha_l y_l K_{kl}$ (the contribution of the fixed variables to point k 's score). Collecting the terms of W that involve α_1, α_2 :

$$W = \alpha_1 + \alpha_2 - \frac{1}{2} K_{11} \alpha_1^2 - \frac{1}{2} K_{22} \alpha_2^2 - s K_{12} \alpha_1 \alpha_2 - y_1 \alpha_1 v_1 - y_2 \alpha_2 v_2 + \text{const.} \quad (2)$$

The equality constraint gives $\alpha_1 = \gamma - s \alpha_2$ with $\gamma = \alpha_1 + s \alpha_2$ (current values). Substituting, W becomes a quadratic in α_2 with second derivative $-(K_{11} + K_{22} - 2K_{12}) = -\eta \leq 0$. Setting $\partial W / \partial \alpha_2 = 0$ and using $v_k = f(\mathbf{x}_k) - b - \alpha_1 y_1 K_{1k} - \alpha_2 y_2 K_{2k}$ to rewrite everything in terms of the current errors gives

$$\alpha_2^{\text{new}} = \alpha_2 + \frac{y_2 (E_1 - E_2)}{\eta}. \quad (3)$$

If $\eta = 0$ (e.g. two identical points) the quadratic is flat along the line and the step is skipped. The b updates follow from requiring $f(\mathbf{x}_1) = y_1$ (if α_1 is free) or $f(\mathbf{x}_2) = y_2$ (if α_2 is free) after the update, writing the new score as the old score plus the change $y_1 \Delta \alpha_1 K_{1k} + y_2 \Delta \alpha_2 K_{2k} + \Delta b$.

5 Solving the Dual with a Generic QP Solver

Optional HW3 Problem 13 solves the dual with a library such as `cvxopt.solvers.qp`, which minimizes $\frac{1}{2}\alpha^T P \alpha + q^T \alpha$ subject to $G\alpha \leq h$ and $A\alpha = b_{eq}$. Maximizing $W(\alpha)$ is minimizing $-W(\alpha)$, so:

$$P = (\mathbf{y}\mathbf{y}^T) \circ K \quad (P_{ij} = y_i y_j K_{ij}), \quad q = -\mathbf{1}, \quad G = \begin{bmatrix} -I \\ I \end{bmatrix}, \quad h = \begin{bmatrix} \mathbf{0} \\ C\mathbf{1} \end{bmatrix}, \quad A = \mathbf{y}^T, \quad b_{eq} = 0, \quad (4)$$

where $\circ$ is the elementwise product, $G\alpha \leq h$ encodes $0 \leq \alpha_i \leq C$ ($2N$ rows), and $A\alpha = 0$ is $\sum_i \alpha_i y_i = 0$. (In `cvxopt` all of these must be `cvxopt.matrix` objects of type double.) The returned α should match SMO's up to the solvers' tolerances; recover b from the free support vectors as in SVM note, §4. On a small two-blob dataset, solving this exact formulation gave α within 10^{-4} of `sklearn`'s.

6 The Full Platt SMO

Platt's original algorithm (HW3 optional Problem 12) adds heuristics for speed:

- **First choice** (i): alternate between a pass over *all* points and repeated passes over only the *non-bound* points ($0 < \alpha_i < C$), which are the ones most likely to still change. Stop when a pass over all points changes nothing.
- **Second choice** (j): pick the j maximizing $|E_i - E_j|$ (the largest step); if that makes no progress, try non-bound points from a random start, then all points.
- **Error cache**: keep E_k for non-bound points and update it after each step instead of recomputing every score from scratch.

7 The SVM as Regularized Loss Minimization (Hinge Loss)

From constraints to a loss. At the optimum of (SVM note, Eq. 11) each slack is as small as its constraint allows, $\xi_i = \max(0, 1 - y_i f(\mathbf{x}_i))$. Substituting, the soft-margin SVM is an *unconstrained* problem. Dividing it by NC (which does not move the minimum) puts it in the same “regularizer + average loss” form as regularized logistic regression:

$$\min_{\mathbf{w}, b} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{i=1}^N \max(0, 1 - y_i(\mathbf{x}_i \mathbf{w} + b)), \quad \lambda = \frac{1}{NC}. \quad (5)$$

The term $\max(0, 1 - m)$, with $m = y_i f(\mathbf{x}_i)$ the point's margin, is the **hinge loss**. Large C is small λ (weak regularization), and vice versa.

Same predictor, different losses. The perceptron, logistic regression, and the SVM all predict with the linear score $f(\mathbf{x}) = \mathbf{x}\mathbf{w} + b$; they differ only in how they charge each training point for its margin m (Figure 1):

- **Perceptron**: $\max(0, -m)$. Only mistakes cost anything; a point barely on the right side is free. No regularizer.
- **Logistic regression**: $\log(1 + e^{-m})$ (the GD/LogReg note's negative log-likelihood, rewritten for labels $y \in \{-1, +1\}$). Smooth, and never exactly zero: every point keeps pulling a little.

- **SVM:** $\max(0, 1 - m)$. Mistakes *and* points inside the margin ($m < 1$) cost; points beyond the margin cost exactly zero. That zero is where the sparsity comes from: those points get $\alpha_i = 0$.

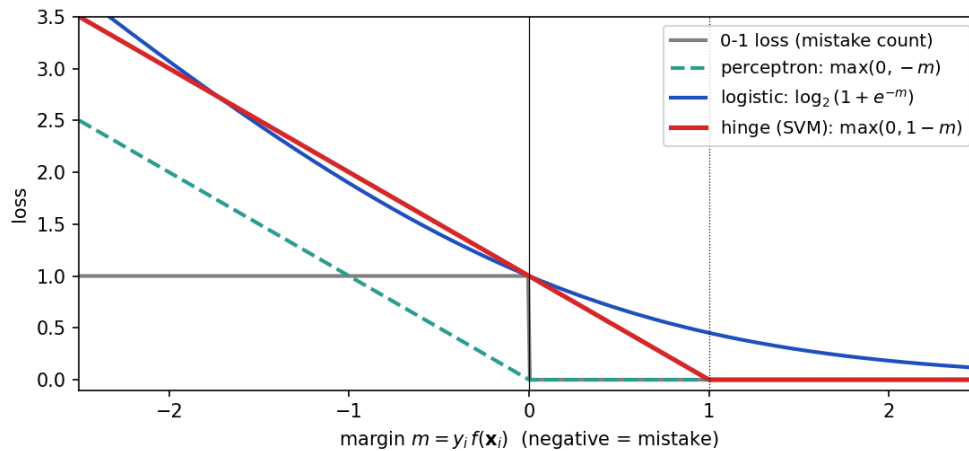


Figure 1: Losses as a function of the margin $m = y_i f(\mathbf{x}_i)$. The logistic loss is scaled by $1/\ln 2$ so that it passes through $(0, 1)$ like the others.

Training by subgradient descent. The hinge has a kink at $m = 1$, so instead of a gradient we use a *subgradient* (any slope that works at the kink; take 0 there). For one point,

$$\frac{\partial}{\partial \mathbf{w}} \max(0, 1 - y_i \mathbf{x}_i^T \mathbf{w}) = \begin{cases} -y_i \mathbf{x}_i^T & \text{if } y_i \mathbf{x}_i^T \mathbf{w} < 1, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Stochastic subgradient descent on (5) with step size $\eta_t = 1/(\lambda t)$ is the **Pegasos** algorithm (Shalev-Shwartz et al., 2007). As in the Regression notes, a dummy feature $x^0 = 1$ folds in the intercept b (this also regularizes b slightly, a harmless difference).

Pegasos: linear SVM by stochastic subgradient descent

Input: training data, $\lambda = 1/(NC)$, number of steps T . Start with $\mathbf{w} = \mathbf{0}$.

For $t = 1, \dots, T$:

- pick a training point i at random; set $\eta_t = 1/(\lambda t)$;
- if $y_i \mathbf{x}_i^T \mathbf{w} < 1$ (inside the margin or a mistake): $\mathbf{w} \leftarrow (1 - \eta_t \lambda) \mathbf{w} + \eta_t y_i \mathbf{x}_i^T$;
- otherwise: $\mathbf{w} \leftarrow (1 - \eta_t \lambda) \mathbf{w}$.

Output $\mathbf{w}$ (or the average of the last half of the iterates, which is a little more stable).

Key point: Compare with the perceptron update $\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i^T$ on mistakes. Pegasos does the same push, but (1) also for correct points inside the margin, and (2) shrinks $\mathbf{w}$ a little at every step: that shrinking is the regularizer $\frac{\lambda}{2} \|\mathbf{w}\|^2$, and it is what produces a large margin.

- **Does it find the SVM?** Yes, up to optimization accuracy: on standardized Spambase ($N = 3220$, $C = 0.01$), 10^5 Pegasos steps (about 30 passes over the data) reach an objective within 0.5% of $\text{SVC}(\text{kernel}='linear')$'s, with cosine similarity 0.995 between the two weight vectors.

- **Why bother?** Each step touches one point, and the cost does not grow with N^2 like a kernel matrix. This is how linear SVMs are trained on very large datasets: `sklearn.svm.LinearSVC` and `SGDClassifier(loss='hinge')`.
- **Linear only, in practice.** A kernelized Pegasos exists, but for kernels use the dual and SMO (SVM note, §5). (Optional HW3 Problem 19 implements Pegasos.)
- The objective (5) also fits the Representer Theorem (Kernels note), another way to see why $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i^T$.

8 Support Vector Regression

The same ideas give a regression method with sparse weights. Replace the hinge by the ε -insensitive loss: residuals smaller than ε are free, larger ones cost linearly,

$$\ell_\varepsilon(y, f(\mathbf{x})) = \max(0, |y - f(\mathbf{x})| - \varepsilon).$$

Picture a **tube** of half-width ε around f : points inside it cost nothing (Figure 2).

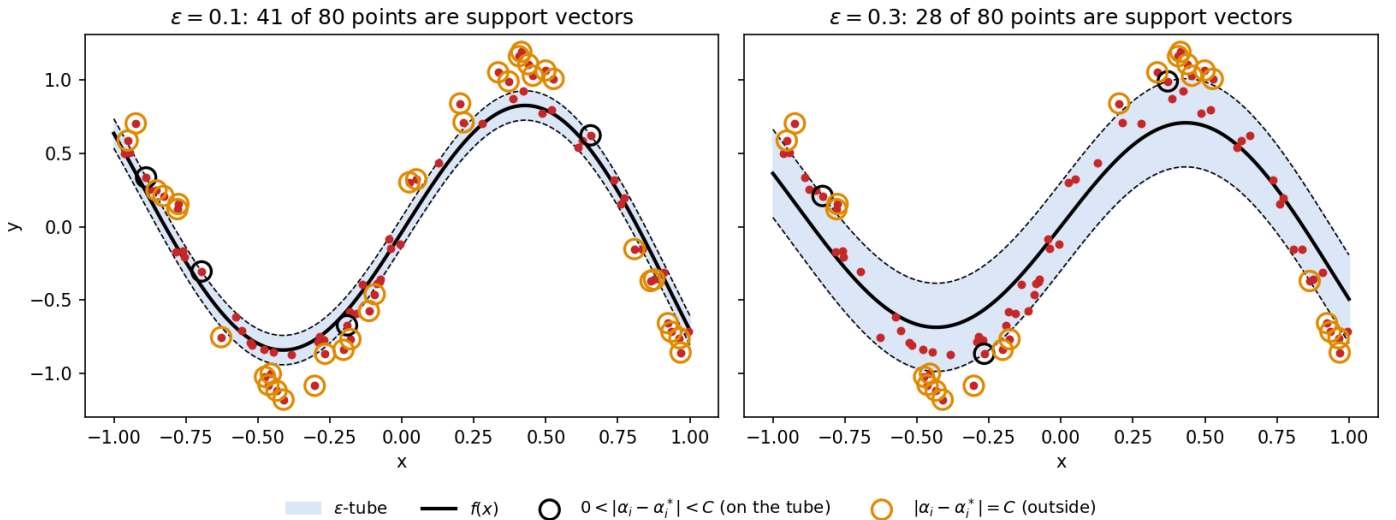


Figure 2: SVR on 80 WAVE points (Gaussian kernel, $\sigma = 1$, $C = 1$). Only points on or outside the tube are support vectors. A wider tube (right) needs fewer support vectors, but the fit is flatter: the peaks are no longer reached.

Primal. Two slacks per point, ξ_i for points above the tube and ξ_i^* for points below:

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

Dual (same Lagrangian recipe as §3, one multiplier α_i or α_i^* per constraint):

$$\max_{\alpha, \alpha^*} -\frac{1}{2} \sum_{i,j} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(\mathbf{x}_i, \mathbf{x}_j) - \varepsilon \sum_i (\alpha_i + \alpha_i^*) + \sum_i y_i (\alpha_i - \alpha_i^*)$$

subject to $\sum_i (\alpha_i - \alpha_i^*) = 0$ and $0 \leq \alpha_i, \alpha_i^* \leq C$. Setting $\partial L / \partial \mathbf{w} = 0$ gives $\mathbf{w} = \sum_i (\alpha_i - \alpha_i^*) \mathbf{x}_i^T$, so the prediction needs only the kernel:

$$f(\mathbf{z}) = \sum_{i=1}^N (\alpha_i - \alpha_i^*) K(\mathbf{x}_i, \mathbf{z}) + b.$$

- **Three types of points** (KKT, as in §2): strictly *inside* the tube: $\alpha_i = \alpha_i^* = 0$, not a support vector; exactly *on* the tube's edge: $0 < |\alpha_i - \alpha_i^*| < C$; *outside*: $|\alpha_i - \alpha_i^*| = C$. A point above the tube has $\alpha_i > 0$ (it pulls f up), a point below has $\alpha_i^* > 0$; never both.
- **Compared with kernel ridge regression** (Kernels note, §8): both predict with a weighted sum of kernel values $K(\mathbf{x}_i, \mathbf{z})$, but kernel ridge regression gives *every* training point a nonzero weight, while SVR gives zero weight to all points inside the tube. Also, the squared loss grows quadratically with the residual, the ε -insensitive loss only linearly, so a few outliers pull SVR less.
- **Choosing ε .** Larger ε : fewer support vectors, but a coarser fit. A good starting point is the noise level of y . On WAVE (300 training points, Gaussian kernel $\sigma = 1$, $C = 1$; noise s.d. ≈ 0.1): $\varepsilon = 0$ makes all 300 points support vectors (test MSE 0.0165); $\varepsilon = 0.1$ keeps 150 (MSE 0.0158); $\varepsilon = 0.3$ keeps 34 (MSE 0.036, the tube is now too wide).
- **Library:** `sklearn.svm.SVR` with `kernel='rbf'`, `gamma=1/\sigma^2`, `C`, and `epsilon`; `dual_coef_` holds $\alpha_i - \alpha_i^*$ for the support vectors and `intercept_` is b . (Optional HW3 Problem 20 compares SVR with kernel ridge regression on WAVE.)

9 Historical: VC Dimension and Margins (superseded)

The original motivation for maximizing the margin came from statistical learning theory. The **VC dimension** of a classifier family is the largest number of points it can label in every possible way. Hyperplanes in $\mathbb{R}^D$ have VC dimension $D + 1$, and generalization bounds grow with it. For hyperplanes with margin γ on data inside a ball of radius R , the bound improves to $\min(D, R^2/\gamma^2) + 1$: a large margin limits capacity regardless of the number of features — the same quantity $(R/\gamma)^2$ as the perceptron's mistake bound. These bounds are usually far too loose to be numerically useful, and modern practice selects C and the kernel by validation; this material (HW3 optional Problem 16) is mainly of historical interest.