A Comprehensive Mathematical Guide to Support Vector Machines

Understanding SVMs from First Principles

January 1, 2026

Abstract

This document provides a complete mathematical treatment of Support Vector Machines (SVMs), one of the most elegant and theoretically grounded algorithms in machine learning. We develop the theory from foundational concepts in functional analysis and optimization, building intuition through geometric interpretation and concrete examples. Our goal is to make the mathematical sophistication of SVMs accessible while maintaining full rigor, showing why these methods work and when they should be applied.

Contents

1 Introduction: What are Support Vector Machines?

1.1 The Big Picture

Imagine you have data points from two different classes scattered on a page, and you want to draw a line separating them. Support Vector Machines solve this problem by finding the "best" separating line—one that not only divides the classes but does so with maximum confidence.

Key insight: The "best" separator is the one that maintains the maximum distance (margin) from both classes. This geometric principle leads to better generalization on unseen data.

1.2 Historical Context

SVMs emerged from statistical learning theory developed by Vladimir Vapnik and Alexey Chervonenkis in the 1960s, reaching their modern form in the 1990s. Three innovations made SVMs revolutionary:

1. **Maximum margin principle**: Geometric foundation for optimal classification
2. **Kernel trick**: Handle non-linear problems without explicit high-dimensional computation
3. **Efficient optimization**: Algorithms like SMO make training computationally feasible

1.3 Why Study SVMs?

Despite the dominance of deep learning, SVMs remain important because they:

- Provide strong theoretical guarantees on generalization
- Work well with limited data
- Are interpretable and have fewer hyperparameters
- Illustrate fundamental principles applicable to other ML methods

2 Mathematical Prerequisites

Before diving into SVMs, we need to establish the mathematical foundations. Don't worry—we'll build intuition alongside formalism.

2.1 Metric Spaces: Measuring Distance

Definition 2.1 (Metric Space)**.** A metric space (X, d) consists of a set X and a distance function $d : X \times X \to \mathbb{R}$ satisfying:

1. $d(x, y) \geq 0$ and $d(x, y) = 0$ if and only if $x = y$ (positivity)
2. $d(x, y) = d(y, x)$ (symmetry)

3. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

Example 2.2 (Euclidean Distance)**.** The most familiar metric is Euclidean distance in $\mathbb{R}^n$:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

This measures the "straight-line" distance between points.

Why this matters: SVMs fundamentally rely on measuring distances between points and decision boundaries. Understanding metrics is crucial.

2.2 Inner Product Spaces: Measuring Angles

Definition 2.3 (Inner Product Space)**.** An inner product space is a vector space V over $\mathbb{R}$ with a map $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ satisfying:

1. $\langle x, y \rangle = \langle y, x \rangle$ (symmetry)
2. $\langle ax + by, z \rangle = a\langle x, z \rangle + b\langle y, z \rangle$ (linearity)
3. $\langle x, x \rangle \geq 0$ with equality if and only if $x = 0$ (positive definiteness)

Example 2.4 (Standard Inner Product)**.** In $\mathbb{R}^n$, the standard inner product is:

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i = x^T y$$

This generalizes the dot product you know from vectors.

Lemma 2.5 (Cauchy-Schwarz Inequality)**.** For any vectors x, y in an inner product space:

$$|\langle x, y \rangle| \leq \|x\| \|y\|$$

where $\|x\| = \sqrt{\langle x, x \rangle}$ is the induced norm.

Proof. Consider the quadratic function $f(t) = \|x + ty\|^2 = \langle x + ty, x + ty \rangle \geq 0$ for all $t \in \mathbb{R}$.

Expanding:

$$f(t) = \langle x, x \rangle + 2t\langle x, y \rangle + t^2 \langle y, y \rangle = \|x\|^2 + 2t\langle x, y \rangle + t^2 \|y\|^2$$

Since $f(t) \geq 0$ for all t, this quadratic has at most one real root, so its discriminant must be non-positive:

$$(2\langle x, y \rangle)^2 - 4\|x\|^2 \|y\|^2 \leq 0$$

Simplifying yields $|\langle x, y \rangle| \leq \|x\| \|y\|$. □

Geometric interpretation: The Cauchy-Schwarz inequality says that the inner product relates to the angle between vectors: $\langle x, y \rangle = \|x\| \|y\| \cos \theta$.

2.3 Hilbert Spaces: Complete Inner Product Spaces

Definition 2.6 (Hilbert Space)**.** A Hilbert space $\mathcal{H}$ is a complete inner product space, meaning every Cauchy sequence in $\mathcal{H}$ converges to a point in $\mathcal{H}$.

Intuition: Completeness means there are no "holes" in the space. This technical property ensures our optimization problems have solutions.

Theorem 2.7 (Riesz Representation Theorem)**.** For any continuous linear functional L on a Hilbert space $\mathcal{H}$, there exists a unique $y \in \mathcal{H}$ such that:

$$L(x) = \langle x, y \rangle \quad \forall x \in \mathcal{H}$$

Why this matters: This theorem is foundational to kernel methods. It tells us that linear functionals (like our classifiers) can be represented as inner products.

3 The Geometry of Linear Classification

Now we get to the heart of SVMs: understanding classification geometrically.

3.1 Hyperplanes: Decision Boundaries

Definition 3.1 (Hyperplane)**.** A hyperplane in $\mathbb{R}^n$ is an affine subspace of dimension $n-1$:

$$\mathcal{H} = \{x \in \mathbb{R}^n : w^T x + b = 0\}$$

where $w \in \mathbb{R}^n \setminus \{0\}$ is the normal vector and $b \in \mathbb{R}$ is the bias term.

Example 3.2 (Hyperplanes in Different Dimensions)**.**

- In $\mathbb{R}^2$: A hyperplane is a line, e.g., $w_1 x_1 + w_2 x_2 + b = 0$
- In $\mathbb{R}^3$: A hyperplane is a plane, e.g., $w_1 x_1 + w_2 x_2 + w_3 x_3 + b = 0$
- In $\mathbb{R}^n$: A hyperplane is an $(n-1)$-dimensional subspace

Classification using hyperplanes: A hyperplane divides space into two half-spaces. We classify a point x by checking which side it's on:

$$f(x) = \text{sign}(w^T x + b) = \begin{cases} +1 & \text{if } w^T x + b > 0 \\ -1 & \text{if } w^T x + b < 0 \end{cases}$$

3.2 Distance from a Point to a Hyperplane

Proposition 3.3 (Signed Distance Formula)**.** The signed distance from a point $x \in \mathbb{R}^n$ to the hyperplane $\mathcal{H} = \{z : w^T z + b = 0\}$ is:

$$d(x, \mathcal{H}) = \frac{w^T x + b}{\|w\|}$$

Proof. Let p be the orthogonal projection of x onto $\mathcal{H}$. Then:

$$x = p + \lambda w$$

for some scalar λ (we move from p along the normal direction).

Since p lies on the hyperplane: $w^T p + b = 0$.

Substituting $p = x - \lambda w$:

$$w^T(x - \lambda w) + b = 0 \implies w^T x + b = \lambda \|w\|^2$$

Therefore:

$$\lambda = \frac{w^T x + b}{\|w\|^2}$$

The distance is $\|\lambda w\| = |\lambda| \|w\| = \frac{|w^T x + b|}{\|w\|}$.

The signed distance (keeping the sign) is $\frac{w^T x + b}{\|w\|}$. □

Example 3.4 (Computing Distance)**.** Consider the line $2x_1 + 3x_2 - 6 = 0$ in $\mathbb{R}^2$ and the point $(5, 4)$.

Here $w = (2, 3)^T$, $b = -6$, so $\|w\| = \sqrt{4 + 9} = \sqrt{13}$.

Distance:

$$d = \frac{2(5) + 3(4) - 6}{\sqrt{13}} = \frac{10 + 12 - 6}{\sqrt{13}} = \frac{16}{\sqrt{13}} \approx 4.44$$

The point is on the positive side of the line.

3.3 The Margin: Core Concept of SVMs

Definition 3.5 (Geometric Margin)**.** For a binary classification dataset $\{(x_i, y_i)\}_{i=1}^n$ where $y_i \in \{-1, +1\}$, the geometric margin of a hyperplane (w, b) is:

$$\gamma = \min_{i=1,\ldots,n} \frac{y_i(w^T x_i + b)}{\|w\|}$$

Interpretation: The margin is the distance from the hyperplane to the nearest point of either class. The factor y_i ensures we measure distance to the correct side.

Lemma 3.6 (Scale Invariance)**.** The geometric margin is invariant under scaling: for any $\alpha > 0$,

$$\gamma(w, b) = \gamma(\alpha w, \alpha b)$$

Proof.

$$\frac{y_i((\alpha w)^T x_i + \alpha b)}{\|\alpha w\|} = \frac{\alpha(y_i(w^T x_i + b))}{\alpha \|w\|} = \frac{y_i(w^T x_i + b)}{\|w\|}$$

□

Why this matters: We can normalize by setting the functional margin to 1 without loss of generality.

4 The Hard-Margin SVM Optimization Problem

4.1 Formulating the Problem

Suppose our data is linearly separable. The SVM finds the maximum-margin hyperplane.

Goal: Maximize the geometric margin $\gamma = \frac{1}{\|w\|}$ subject to correct classification.

Using scale invariance, we can require:

$$y_i(w^T x_i + b) \geq 1 \quad \forall i$$

This is called the **canonical form**: the closest points satisfy $y_i(w^T x_i + b) = 1$.

Maximizing $\frac{1}{\|w\|}$ is equivalent to minimizing $\|w\|$, or equivalently $\frac{1}{2}\|w\|^2$ (for computational convenience).

Hard-Margin SVM: Primal Problem

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n \end{aligned}$$

4.2 Example: Two-Dimensional Case

Example 4.1 (SVM in 2D)**.** Consider four points:

- Class $+1$: $(3, 3)$, $(4, 4)$
- Class -1: $(1, 1)$, $(2, 1)$

We want to find $w = (w_1, w_2)^T$ and b such that:

$$\begin{aligned} 3w_1 + 3w_2 + b &\geq 1 \\ 4w_1 + 4w_2 + b &\geq 1 \\ -(w_1 + w_2 + b) &\geq 1 \\ -(2w_1 + w_2 + b) &\geq 1 \end{aligned}$$

while minimizing $w_1^2 + w_2^2$.

By symmetry and inspection, we can verify that $w = (0, 1)^T$ and $b = -2$ gives:

- $(3, 3)$: $0 + 3 - 2 = 1$
- $(4, 4)$: $0 + 4 - 2 = 2 \geq 1$
- $(1, 1)$: $-(0 + 1 - 2) = 1$
- $(2, 1)$: $-(0 + 1 - 2) = 1$

The separating line is $x_2 = 2$, with margin $\gamma = 1$.

5 Optimization Theory for SVMs

5.1 Convex Optimization Primer

Definition 5.1 (Convex Function)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^n$ and $\theta \in [0, 1]$:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

Geometric intuition: A function is convex if the line segment between any two points on its graph lies above the graph.

Why SVMs work well: The SVM optimization problem is convex with linear constraints, guaranteeing a unique global optimum.

5.2 Lagrangian Duality

To solve the constrained optimization problem, we introduce Lagrange multipliers.

Definition 5.2 (Lagrangian for Hard-Margin SVM)**.**

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

where $\alpha_i \geq 0$ are Lagrange multipliers.

Intuition: We subtract penalty terms for constraint violations, weighted by α_i.

5.3 KKT Conditions

Theorem 5.3 (Karush-Kuhn-Tucker Conditions)**.** At the optimal solution (w^*, b^*, α^*), the following conditions hold:

Stationarity:

$$\nabla_w \mathcal{L} = w^* - \sum_{i=1}^{n} \alpha_i^* y_i x_i = 0 \tag{1}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i^* y_i = 0 \tag{2}$$

Primal feasibility:

$$y_i(w^{*T} x_i + b^*) \geq 1, \quad \forall i$$

Dual feasibility:

$$\alpha_i^* \geq 0, \quad \forall i$$

Complementary slackness:

$$\alpha_i^* [y_i(w^{*T} x_i + b^*) - 1] = 0, \quad \forall i$$

Key insight from complementary slackness: Either $\alpha_i^* = 0$ or $y_i(w^{*T} x_i + b^*) = 1$.

Points with $\alpha_i^* > 0$ lie exactly on the margin boundary. These are the **support vectors**!

5.4 The Dual Problem

From the stationarity condition: $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$.

Substituting into the Lagrangian and simplifying yields:

Hard-Margin SVM: Dual Problem

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

Why the dual is useful:

1. Depends only on inner products $x_i^T x_j$ (crucial for the kernel trick!)
2. Often easier to solve than the primal
3. Directly identifies support vectors

5.5 Making Predictions

Once we have α^*, we can classify new points:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* y_i x_i^T x + b^* \right)$$

But only support vectors (where $\alpha_i^* > 0$) contribute:

$$f(x) = \text{sign}\left(\sum_{i \in SV} \alpha_i^* y_i x_i^T x + b^* \right)$$

To find b^*, use any support vector x_s:

$$b^* = y_s - \sum_{i \in SV} \alpha_i^* y_i x_i^T x_s$$

6 Soft-Margin SVMs: Handling Noisy Data

Real data is rarely perfectly separable. We need to allow some misclassifications.

6.1 The Soft-Margin Formulation

Introduce **slack variables** $\xi_i \geq 0$ to measure constraint violations:

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

Soft-Margin SVM: Primal Problem

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

Interpretation:

- C is a hyperparameter balancing margin size vs. training error
- Large C: prioritize correct classification (risk overfitting)
- Small C: prioritize large margin (tolerate more errors)

6.2 Geometric Interpretation of Slack Variables

- $\xi_i = 0$: point is correctly classified beyond the margin
- $0 < \xi_i < 1$: point is correctly classified but inside the margin
- $\xi_i = 1$: point is exactly on the decision boundary
- $\xi_i > 1$: point is misclassified

6.3 The Soft-Margin Dual

Soft-Margin SVM: Dual Problem

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \end{aligned}$$

Note: The only difference from hard-margin is the box constraint $\alpha_i \leq C$.

7 The Kernel Trick: Non-Linear Classification

Most real problems aren't linearly separable. The kernel trick is the elegant solution.

7.1 Feature Space Mapping

Idea: Map data to a higher-dimensional space where it becomes linearly separable.

Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a feature map to a (possibly infinite-dimensional) Hilbert space.

In feature space, our classifier becomes:

$$f(x) = \text{sign}(w^T \phi(x) + b)$$

The dual problem becomes:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$$

7.2 The Kernel Function

Definition 7.1 (Kernel Function)**.** A kernel is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle$$

for some feature map ϕ into a Hilbert space $\mathcal{H}$.

The magic: We can compute inner products in high-dimensional space without explicitly computing $\phi(x)$!

7.3 Common Kernels

Example 7.2 (Popular Kernel Functions)**. 1. Linear Kernel:**

$$K(x, x') = x^T x'$$

This is just the standard SVM (no transformation).

2. Polynomial Kernel:

$$K(x, x') = (x^T x' + c)^d$$

Implicitly maps to a space of degree-d polynomials.

3. Gaussian RBF Kernel:

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2} \right)$$

Maps to an infinite-dimensional space! Yet computable in closed form.

4. Sigmoid Kernel:

$$K(x, x') = \tanh(\kappa x^T x' + c)$$

Resembles neural network activation.

7.4 Example: Polynomial Kernel Expansion

Example 7.3 (Explicit Polynomial Feature Map)**.** Consider $x, x' \in \mathbb{R}^2$ with $d = 2$ and $c = 0$.

The polynomial kernel is: $K(x, x') = (x^T x')^2$.

Let's see the explicit feature map. If $x = (x_1, x_2)^T$:

$$(x^T x')^2 = (x_1 x_1' + x_2 x_2')^2 = x_1^2 {x_1'}^2 + 2x_1 x_2 x_1' x_2' + x_2^2 {x_2'}^2$$

This equals:

$$\langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2), ({x_1'}^2, \sqrt{2}x_1' x_2', {x_2'}^2) \rangle$$

So $\phi(x) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T$ maps $\mathbb{R}^2 \to \mathbb{R}^3$.

But computing $K(x, x')$ directly is faster than computing ϕ explicitly!

7.5 Valid Kernels: Mercer's Theorem

Theorem 7.4 (Mercer's Theorem (Simplified))**.** A symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a valid kernel if and only if for any finite set of points $\{x_1, \ldots, x_n\}$, the Gram matrix:

$$\mathbf{K} = [K(x_i, x_j)]_{i,j=1}^n$$

is positive semidefinite.

Checking positive semidefiniteness: For all vectors $\alpha \in \mathbb{R}^n$:

$$\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(x_i, x_j) \geq 0$$

7.6 Building New Kernels

Theorem 7.5 (Kernel Combinations)**.** If K_1 and K_2 are valid kernels, then so are:

1. $K(x, x') = K_1(x, x') + K_2(x, x')$ (sum)
2. $K(x, x') = cK_1(x, x')$ for $c > 0$ (positive scaling)
3. $K(x, x') = K_1(x, x') \cdot K_2(x, x')$ (product)
4. $K(x, x') = f(x)f(x')$ for any function f (rank-one kernel)

8 Reproducing Kernel Hilbert Spaces (RKHS)

8.1 The RKHS Framework

Definition 8.1 (RKHS)**.** A Reproducing Kernel Hilbert Space $\mathcal{H}$ is a Hilbert space of functions where point evaluation functionals are continuous:

$$\exists C_x > 0 : |f(x)| \leq C_x \|f\|_{\mathcal{H}} \quad \forall f \in \mathcal{H}$$

Key property: For each x, there exists a representer $K_x \in \mathcal{H}$ such that:

$$f(x) = \langle f, K_x \rangle_{\mathcal{H}}$$

And $K(x, x') = \langle K_x, K_{x'} \rangle_{\mathcal{H}}$ is the reproducing kernel.

8.2 The Representer Theorem

Theorem 8.2 (Representer Theorem)**.** The solution to the regularized risk minimization:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2$$

has the form:

$$f^*(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

Why this is profound: Despite optimizing over an infinite-dimensional space, the solution lies in the finite-dimensional span of $\{K(\cdot, x_i)\}_{i=1}^n$.

Proof sketch. Decompose $f = f_\parallel + f_\perp$ where $f_\parallel \in \text{span}\{K(\cdot, x_i)\}$ and $f_\perp \perp f_\parallel$.

Note that $f(x_i) = f_\parallel(x_i) + f_\perp(x_i) = f_\parallel(x_i)$ since $f_\perp(x_i) = \langle f_\perp, K_{x_i} \rangle = 0$.

Thus the loss term depends only on $f_\parallel$, while:

$$\|f\|^2 = \|f_\parallel\|^2 + \|f_\perp\|^2 \geq \|f_\parallel\|^2$$

So we can always improve or maintain the objective by setting $f_\perp = 0$. □

9 Statistical Learning Theory

9.1 The Learning Problem

Setup: We have:

- Unknown joint distribution $P(X, Y)$ over input-output pairs
- Training sample $S = \{(x_i, y_i)\}_{i=1}^n$ drawn i.i.d. from P
- Hypothesis class $\mathcal{F}$ (e.g., linear classifiers)
- Loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

Definition 9.1 (Expected Risk)**.** The expected risk (true error) of a hypothesis f is:

$$R(f) = \mathbb{E}_{(X,Y)\sim P}[L(Y, f(X))] = \int L(y, f(x))\, dP(x, y)$$

Definition 9.2 (Empirical Risk)**.** The empirical risk (training error) is:

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))$$

Fundamental question: When does $\hat{R}_n(f) \approx R(f)$ for all $f \in \mathcal{F}$?

9.2 VC Dimension

Definition 9.3 (Shattering)**.** A set of points $\{x_1, \ldots, x_m\}$ is **shattered** by hypothesis class $\mathcal{F}$ if for every possible labeling $\{y_1, \ldots, y_m\} \in \{-1, +1\}^m$, there exists $f \in \mathcal{F}$ that realizes this labeling: $f(x_i) = y_i$ for all i.

Definition 9.4 (VC Dimension)**.** The VC dimension of $\mathcal{F}$, denoted $\text{VC}(\mathcal{F})$, is the maximum number of points that can be shattered by $\mathcal{F}$.

Theorem 9.5 (VC Dimension of Linear Classifiers)**.** The VC dimension of hyperplanes in $\mathbb{R}^d$ is $d + 1$.

Proof sketch. **Upper bound:** Any $d + 2$ points in $\mathbb{R}^d$ are affinely dependent, so cannot be shattered.

Lower bound: Consider $d + 1$ points in general position (e.g., vertices of a simplex plus origin). We can show any labeling is achievable by constructing the appropriate hyperplane.

For example, in $\mathbb{R}^2$: the three points $(0, 0), (1, 0), (0, 1)$ can be shattered by lines. Any of the $2^3 = 8$ labelings can be achieved. □

9.3 Generalization Bounds

Theorem 9.6 (Uniform Convergence Bound)**.** With probability at least $1 - \delta$:

$$\sup_{f \in \mathcal{F}} |R(f) - \hat{R}_n(f)| \leq \sqrt{\frac{h(\log(2n/h) + 1) - \log(\delta/4)}{n}}$$

where $h = \text{VC}(\mathcal{F})$.

Interpretation:

- Generalization error decreases as $O(1/\sqrt{n})$
- Higher VC dimension requires more training data
- This bound applies *uniformly* to all $f \in \mathcal{F}$

9.4 Why Maximum Margin Helps

Theorem 9.7 (Margin-Based VC Bound)**.** For linear classifiers with margin γ on data in a ball of radius R, the VC dimension is bounded by:

$$h \leq \min\left\{d + 1, \left\lceil \frac{R^2}{\gamma^2} \right\rceil\right\}$$

Key insight: By maximizing the margin, SVMs effectively reduce the VC dimension, leading to better generalization—even in high-dimensional spaces!

10 Practical Implementation

10.1 Sequential Minimal Optimization (SMO)

The dual problem requires solving a quadratic program with n variables. For large n, this is expensive.

SMO insight: Optimize two variables at a time while fixing all others.

Algorithm sketch:

1. Select two variables α_i, α_j (often choosing those that violate KKT conditions most)
2. Solve the two-variable subproblem analytically (quadratic problem in 1D after eliminating one variable using $\sum \alpha_k y_k = 0$)
3. Update α_i, α_j
4. Repeat until convergence

Advantages:

- No matrix inversion required
- Memory efficient (doesn't store full Gram matrix)
- Fast convergence in practice

10.2 Computing the Bias Term

After finding α^*, compute b using any support vector x_s with $0 < \alpha_s < C$:

$$b^* = y_s - \sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x_s)$$

Robust approach: Average over all support vectors:

$$b^* = \frac{1}{|SV|} \sum_{s \in SV} \left(y_s - \sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x_s) \right)$$

10.3 Hyperparameter Selection

Key hyperparameters:

- C: regularization parameter (soft-margin trade-off)
- Kernel type: linear, polynomial, RBF, etc.
- Kernel parameters: e.g., σ for RBF, degree d for polynomial

Selection method: Use cross-validation

1. Split data into k folds
2. For each hyperparameter configuration:
   - Train on $k-1$ folds, validate on remaining fold
   - Repeat for all folds, average performance
3. Select configuration with best average validation performance

11 Advanced Topics

11.1 Multi-Class SVMs

SVMs are inherently binary classifiers. For $K > 2$ classes:

One-vs-Rest (OvR):

- Train K binary SVMs
- Classifier k: class k vs. all others
- Predict: $\arg\max_k f_k(x)$

One-vs-One (OvO):

- Train $\binom{K}{2}$ binary SVMs
- One for each pair of classes
- Predict: majority vote

11.2 SVMs for Regression (SVR)

Instead of classification, predict continuous values using ϵ-insensitive loss:

$$L_\epsilon(y, f(x)) = \max(0, |y - f(x)| - \epsilon)$$

This creates a "tube" of width 2ϵ around the regression function—errors within the tube are not penalized.

11.3 Imbalanced Data

When one class heavily outweighs another, use class-weighted SVMs:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C_+ \sum_{i:y_i=+1} \xi_i + C_- \sum_{i:y_i=-1} \xi_i$$

Typically set $C_+ > C_-$ to penalize misclassification of the minority class more heavily.

11.4 Online and Large-Scale SVMs

For massive datasets, exact SVM training is infeasible. Approaches include:

Stochastic Gradient Descent: Instead of solving the dual, directly optimize the primal using subgradients:

$$w_{t+1} = w_t - \eta_t \left(\lambda w_t - y_i x_i \mathbb{I}[y_i(w_t^T x_i) < 1]\right)$$

Core Vector Machines: Approximate the minimum enclosing ball in feature space, reducing effective training set size.

12 Practical Examples

12.1 Example 1: Text Classification

Problem: Classify emails as spam or not spam.

Approach:

1. **Feature extraction:** Use bag-of-words or TF-IDF vectors (high-dimensional, $d \approx 10,000$)
2. **Kernel choice:** Linear kernel (efficient, works well for text)
3. **Why SVM works:** Text data often linearly separable in high dimensions

Training:

- Dataset: 5,000 emails (3,000 ham, 2,000 spam)
- Use class weighting: $C_{\text{spam}} = 1.5 \times C_{\text{ham}}$
- Cross-validate to find $C \in \{0.01, 0.1, 1, 10\}$

12.2 Example 2: Image Classification

Problem: Handwritten digit recognition (0-9).

Approach:

1. **Features:** Raw pixel values ($28 \times 28 = 784$ dimensions) or extracted features (HOG, SIFT)
2. **Kernel:** RBF kernel with σ chosen by cross-validation
3. **Multi-class:** One-vs-Rest (train 10 binary SVMs)

Why RBF kernel:

- Captures non-linear relationships between pixels
- Can model complex decision boundaries
- Performs well even with moderate training data

12.3 Example 3: Bioinformatics

Problem: Protein fold classification from amino acid sequences.

Approach:

1. **Custom kernel:** String kernel measuring subsequence similarity
2. **Why specialized kernel:** Captures biological properties (motifs, conserved regions)

String kernel example:

$$K(s,t) = \sum_{u \in \Sigma^p} \#(u \text{ in } s) \times \#(u \text{ in } t)$$

where Σ^p is the set of all length-p subsequences.

13 When to Use SVMs

13.1 SVMs Excel When:

1. **High-dimensional data:** SVMs work well when $d \gg n$ (features exceed samples)
2. **Clear margin of separation:** Classes are well-separated
3. **Limited training data:** SVMs generalize well with smaller datasets
4. **Need for interpretability:** Support vectors provide insight into decision boundary
5. **Kernel structure available:** Domain knowledge suggests good kernel choice

13.2 Consider Alternatives When:

1. **Very large datasets:** $n > 100,000$ (neural networks scale better)
2. **Noisy labels:** Probabilistic methods like logistic regression may be preferable
3. **Multi-class with many classes:** Decision trees or neural networks are more natural
4. **Need probability estimates:** SVMs don't naturally output probabilities (though Platt scaling can be applied)

14 Connections to Other Methods

14.1 Logistic Regression vs. SVM

Both solve similar problems but differ in loss functions:

Logistic Regression:

$$\min_w \sum_{i=1}^{n} \log(1 + e^{-y_i w^T x_i}) + \lambda \|w\|^2$$

SVM:

$$\min_w \sum_{i=1}^{n} \max(0, 1 - y_i w^T x_i) + \lambda \|w\|^2$$

Key difference: Hinge loss (SVM) has sparse solutions (only support vectors matter); log loss (LR) considers all points.

14.2 Relationship to Neural Networks

Similarities:

- Both learn non-linear decision boundaries
- Both can use gradient-based optimization

- RBF kernel SVM resembles one-hidden-layer RBF network

Differences:

- SVMs: convex optimization, global optimum guaranteed
- Neural networks: non-convex, multiple local minima, but more flexible
- SVMs: theory-driven; NNs: empirically-driven

15 Summary and Key Takeaways

15.1 Core Principles

1. **Maximum margin:** Choose the separator farthest from both classes
2. **Support vectors:** Only boundary points matter for classification
3. **Kernel trick:** Implicitly work in high dimensions without explicit computation
4. **Convex optimization:** Guaranteed to find global optimum
5. **Theoretical guarantees:** VC theory provides generalization bounds

15.2 The Mathematics Behind SVMs

SVMs beautifully unify multiple mathematical areas:

- **Geometry:** Hyperplanes, margins, distance
- **Functional analysis:** Hilbert spaces, RKHS, Representer theorem
- **Optimization:** Convex programming, Lagrangian duality, KKT conditions
- **Statistics:** VC theory, generalization bounds, risk minimization

15.3 Practical Wisdom

- **Start simple:** Try linear SVM first; add complexity if needed
- **Scale features:** Normalize data before training
- **Cross-validate:** Always tune C and kernel parameters
- **Interpret support vectors:** They tell you which examples are "difficult"
- **Consider computational cost:** Training is $O(n^2)$ to $O(n^3)$; for very large n, consider alternatives

16 Conclusion

Support Vector Machines represent one of the most mathematically elegant and practically effective machine learning algorithms. Their development synthesized deep results from functional analysis, optimization theory, and statistical learning theory into a coherent framework with strong theoretical guarantees and excellent empirical performance.

The key insights—maximum margin principle, kernel methods, and convex optimization—extend far beyond SVMs themselves, influencing modern machine learning broadly. While deep learning has captured much recent attention, the mathematical foundations of SVMs remain essential for understanding learning algorithms and continue to inspire new methods.

By mastering SVMs, you gain:

- A principled approach to classification grounded in theory
- Intuition about the geometry of learning
- Tools for working with high-dimensional and non-linear data
- A foundation for understanding more advanced methods

The journey from simple linear separators to kernel methods in infinite-dimensional spaces illustrates the power of mathematical abstraction in solving practical problems. SVMs remind us that elegant theory and practical effectiveness can go hand in hand.

A Computational Complexity Summary

Operation	**Time Complexity**	**Space Complexity**
Training (standard QP)	$O(n^3)$	$O(n^2)$
Training (SMO)	$O(n^2)$ to $O(n^3)$	$O(n)$
Prediction	$O(\|SV\| \cdot d)$	$O(\|SV\| \cdot d)$
Kernel computation	$O(d)$	$O(1)$

where n is training samples, d is feature dimension, $|SV|$ is number of support vectors.

B Common Kernels Reference

Kernel	**Formula**	**Use Case**
Linear	$K(x, x') = x^T x'$	High-dim data, text
Polynomial	$K(x, x') = (x^T x' + c)^d$	Structured features
RBF/Gaussian	$K(x, x') = \exp(-\|x - x'\|^2/2\sigma^2)$	General purpose
Sigmoid	$K(x, x') = \tanh(\kappa x^T x' + c)$	Neural network-like

C Further Reading

For deeper exploration:

- **Vapnik (1995)**: *The Nature of Statistical Learning Theory*—foundational theory

- **Schölkopf & Smola (2002)**: *Learning with Kernels*—comprehensive kernel methods
- **Cristianini & Shawe-Taylor (2000)**: *An Introduction to SVMs*—accessible introduction
- **Burges (1998)**: Tutorial paper—excellent practical overview