Support Vector Machines

Luu Minh Sao Khue

December 9, 2025

1. Introduction

Support Vector Machines (SVMs) are supervised learning methods that can be used for both classification and regression. The regression version is known as Support Vector Regression (SVR), while the classical SVM formulation is designed for classification tasks. In this lecture, we focus only on the binary classification setting, because it highlights the core geometric and mathematical ideas behind SVMs—such as the separating hyperplane, the margin, and the dual formulation—which form the foundation for understanding all other SVM variants.

The main idea is intuitive: SVMs try to

- find a line (in 2D), a plane (in 3D), or a hyperplane (in higher dimensions)
- that separates the two classes,
- and leaves the largest possible "gap" (margin) between them.

In the following sections, we build the SVM formulation step by step. We start with the intuition of the decision boundary, then derive the optimization problem for the linear SVM, introduce the Lagrangian and its dual form, and finally explain the kernel trick and how predictions are made.

2. Problem Setup

We consider a binary classification problem with N training examples. Each example has a feature vector and a label:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N),$$

where:

- $x_i \in \mathbb{R}^d$ is the feature vector of example i (with d features),
- $y_i \in \{+1, -1\}$ is the class label of example i.

Our goal is to learn a rule that, given a new point x, predicts its label $\hat{y} \in \{+1, -1\}$.

3. Intuition: Decision Boundary and Margin

Decision Boundary

SVMs use a linear decision function of the form

$$f(x) = w^\top x + b,$$

where:

- $w \in \mathbb{R}^d$ is called the *weight vector* or *normal vector*,
- $b \in \mathbb{R}$ is called the *bias* or *intercept*,
- $w^\top x$ is the dot product $w^\top x = \sum_{j=1}^{d} w_j x_j$.

The *decision boundary* is the set of points x such that

$$f(x) = 0 \quad \Longleftrightarrow \quad w^\top x + b = 0.$$

These points form a hyperplane in $\mathbb{R}^d$. In 2D, this is a line; in 3D, it is a plane.

We use the sign of $f(x)$ to classify:

$$\hat{y} = \begin{cases} +1, & \text{if } f(x) \geq 0, \\ -1, & \text{if } f(x) < 0. \end{cases}$$

Margin

The *margin* is the distance between the decision boundary and the closest training points from each class.

Intuitively:

- We want the decision boundary to be as far as possible from all training points.
- If the boundary is far from the points, small changes in the data are less likely to change their class, which helps generalization.

SVM explicitly maximizes this margin.

3.1 Geometric Setup: Margin Hyperplanes

We define two parallel hyperplanes:

$$\begin{aligned} w^\top x + b &= +1 \quad \text{(positive margin)}, \\ w^\top x + b &= -1 \quad \text{(negative margin)}. \end{aligned}$$

The decision boundary is exactly in the middle:

$$w^\top x + b = 0.$$

We want:

- all points with label $+1$ to be on or beyond the positive side,
- all points with label -1 to be on or beyond the negative side.

This leads to the following constraints:

$$y_i = +1 \;\Rightarrow\; w^\top x_i + b \geq 1,$$
$$y_i = -1 \;\Rightarrow\; w^\top x_i + b \leq -1.$$

We can combine these two cases using $y_i \in \{+1, -1\}$:

$$y_i\big(w^\top x_i + b\big) \geq 1 \quad \text{for all } i = 1, \ldots, N.$$

4. Margin Width in Terms of $\|w\|$

The margin is the distance between the two parallel hyperplanes

$$w^\top x + b = +1 \qquad \text{and} \qquad w^\top x + b = -1.$$

To see how the margin width is computed, consider:

- a point x_+ lying on the positive-margin hyperplane,
- a point x_- lying on the negative-margin hyperplane.

These satisfy:

$$w^\top x_+ + b = 1, \qquad w^\top x_- + b = -1.$$

We calculate the difference between the two sides by subtracting the negative-side expression from the positive-side expression:

$$\big(w^\top x_+ + b\big) - \big(w^\top x_- + b\big) = 1 - (-1).$$

The bias terms cancel, giving:

$$w^\top (x_+ - x_-) = 2.$$

Geometrically, the vector $(x_+ - x_-)$ points from a point on the negative-margin hyperplane to a point on the positive-margin hyperplane. This vector is not necessarily perpendicular to the hyperplanes, so to find the actual distance (the margin width), we must project it onto the direction that *is* perpendicular to both hyperplanes.

The normal direction of the hyperplanes is given by w. The *unit vector* in this direction is:

$$\hat{w} = \frac{w}{\|w\|}.$$

To obtain the margin width, we take the difference vector and project it onto the unit normal vector:

$$\text{margin} = \big(x_+ - x_-\big)^\top \hat{w} = \frac{w^\top (x_+ - x_-)}{\|w\|}.$$

Using the earlier identity $w^\top (x_+ - x_-) = 2$, we obtain the final margin formula:

$$\text{margin} = \frac{2}{\|w\|}.$$

Thus, the width of the margin becomes larger when $\|w\|$ is smaller. Maximizing the margin is therefore equivalent to minimizing $\|w\|$, which explains why the SVM objective minimizes $\frac{1}{2}\|w\|^2$.

5. Primal Optimization Problem (Hard-Margin Linear SVM)

We can now write the SVM optimization problem.

We want to:

- maximize the margin $\frac{2}{\|w\|}$, which is equivalent to minimizing $\frac{1}{2}\|w\|^2$,
- satisfy the classification constraints $y_i(w^\top x_i + b) \geq 1$ for all i.

This gives the *primal problem*:

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

Here:

- the objective $\frac{1}{2}\|w\|^2$ encourages a large margin,
- the constraints ensure that all training points are correctly classified and lie on or outside the margin boundaries.

This is the *hard-margin* SVM, which assumes that the data are perfectly separable by a hyperplane.

6. Lagrange Multipliers and the Lagrangian

Solving a constrained optimization problem directly can be difficult. We use *Lagrange multipliers* to handle the constraints. Using Lagrange multipliers allows us to incorporate the constraints into a single objective function, so that instead of solving a constrained problem directly, we can optimize an unconstrained function whose saddle point automatically satisfies the constraints.

For each constraint

$$y_i\big(w^\top x_i + b\big) \geq 1,$$

we introduce a non-negative multiplier $\alpha_i \geq 0$. These α_i are called *Lagrange multipliers*.

We form the *Lagrangian*:

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

where

- w and b are the primal variables,
- $\alpha = (\alpha_1, \ldots, \alpha_N)$ are the dual variables (Lagrange multipliers),
- the term $\big[y_i(w^\top x_i + b) - 1\big]$ measures how well constraint i is satisfied.

The idea is:

- If a constraint is violated, the term $y_i(w^\top x_i + b) - 1$ becomes negative, and since $\alpha_i \geq 0$, the Lagrangian L increases, making the solution worse.
- At the optimum, constraints that matter will have $\alpha_i > 0$; those that are not tight will have $\alpha_i = 0$.

We seek a saddle point of L:

$$\min_{w,b} \max_{\alpha \geq 0} L(w, b, \alpha).$$

This saddle point means that we first choose (w, b) to minimize the Lagrangian while the multipliers α try to maximize it, and the point at which these opposing effects balance corresponds exactly to a solution that satisfies both the objective and the original constraints.

7. Stationarity Conditions: Solving for w and b

We first minimize the Lagrangian with respect to w and b by taking partial derivatives and setting them to zero.

Derivative with respect to w

The derivative of L with respect to w is:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i.$$

Explanation:

- The derivative of $\frac{1}{2}\|w\|^2$ with respect to w is w.
- The derivative of $-\sum_{i=1}^{N} \alpha_i y_i (w^\top x_i)$ with respect to w is $-\sum_{i=1}^{N} \alpha_i y_i x_i$.
- The remaining terms do not depend on w and have derivative 0.

Setting this derivative to zero gives:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i.$$

This means that the optimal weight vector w is a linear combination of the training points, weighted by $\alpha_i y_i$.

Derivative with respect to b

The derivative of L with respect to b is:

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i.$$

We obtain this because only the term $-\sum_{i=1}^{N} \alpha_i y_i b$ depends on b, and its derivative with respect to b is $-\sum_{i=1}^{N} \alpha_i y_i$.

Setting this derivative to zero gives:

$$\sum_{i=1}^{N} \alpha_i y_i = 0.$$

This is a constraint on the multipliers α_i.

8. The Dual Problem

We now substitute

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

into the Lagrangian L to obtain a function that depends only on α.

First, compute $\|w\|^2$:

$$\|w\|^2 = w^\top w = \left(\sum_{i=1}^{N} \alpha_i y_i x_i\right)^\top \left(\sum_{j=1}^{N} \alpha_j y_j x_j\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i^\top x_j).$$

After some algebra (omitted for brevity), the dual objective becomes:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i^\top x_j).$$

The dual optimization problem is:

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

This is called the *dual problem*. It depends only on the multipliers α_i and the dot products $x_i^\top x_j$.

At the optimum:

- Many α_i become zero.
- Only points with $\alpha_i > 0$ influence the classifier.
- These points are called *support vectors*.

8.1 Recovering w and b and Making Predictions

After solving the dual problem and obtaining the optimal $\alpha_i^\star$, we can recover:

Weight vector

$$w = \sum_{i=1}^{N} \alpha_i^\star y_i x_i.$$

Bias term

To compute b, we can use any support vector (x_k, y_k) with $\alpha_k^\star > 0$. For such a point, the constraint is tight:

$$y_k\big(w^\top x_k + b\big) = 1.$$

Thus:

$$b = y_k - w^\top x_k = y_k - \sum_{i=1}^{N} \alpha_i^\star y_i (x_i^\top x_k).$$

Prediction for a new point

Given a new point x, the decision function is:

$$f(x) = w^\top x + b = \sum_{i=1}^{N} \alpha_i^\star y_i (x_i^\top x) + b.$$

We predict:

$$\hat{y} = \operatorname{sign}\big(f(x)\big) = \begin{cases} +1, & \text{if } f(x) \geq 0, \\ -1, & \text{if } f(x) < 0. \end{cases}$$

In practice, the sum over i is only over support vectors (those with $\alpha_i^\star > 0$), which makes prediction efficient.

9. Kernel Trick: Non-Linear SVMs

So far, we have considered a *linear* SVM: the decision boundary is a hyperplane in the original feature space.

If the data are not linearly separable, we may want a more complex, curved decision boundary. One way is to map the data into a higher-dimensional feature space.

Feature Mapping

Suppose we have a mapping

$$\phi : \mathbb{R}^d \to \mathbb{R}^D,$$

which transforms the original feature vector x into a new feature vector $\phi(x)$ in a (possibly much higher) dimensional space.

If we run a linear SVM in this new space, the dual problem will depend on inner products of the form

$$\phi(x_i)^\top \phi(x_j).$$

However, explicitly computing $\phi(x)$ can be expensive or even impossible if D is very large or infinite.

Kernel Function

A *kernel function* K is defined as

$$K(x_i, x_j) = \phi(x_i)^\top \phi(x_j).$$

The key idea of the *kernel trick* is that we do not need to know $\phi(x)$ explicitly. We only need to be able to compute $K(x_i, x_j)$.

To obtain a kernelized SVM, we replace each dot product $x_i^\top x_j$ in the dual problem with $K(x_i, x_j)$.

The dual problem becomes:

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

The decision function for a new point x is:

$$f(x) = \sum_{i=1}^{N} \alpha_i^\star y_i K(x_i, x) + b,$$

and the prediction is again

$$\hat{y} = \operatorname{sign}\big(f(x)\big).$$

Common Kernels

Some common choices of kernel functions are:

- **Linear kernel:**
$$K(x_i, x_j) = x_i^\top x_j.$$
This corresponds to the original linear SVM (no feature mapping).

- **Polynomial kernel:**
$$K(x_i, x_j) = \big(x_i^\top x_j + c\big)^d,$$
where $c \geq 0$ is a constant and d is the degree of the polynomial. This allows polynomial decision boundaries.

- **Radial Basis Function (RBF) kernel:**
$$K(x_i, x_j) = \exp\big(-\gamma \|x_i - x_j\|^2\big),$$
where $\gamma > 0$ controls how quickly the similarity decreases with distance. This kernel can model very flexible, non-linear boundaries.

- **Sigmoid kernel:**
$$K(x_i, x_j) = \tanh(a\, x_i^\top x_j + c),$$
where a and c are parameters. This kernel is related to neural network activations.

10. Summary

- SVMs search for a hyperplane $w^\top x + b = 0$ that separates the two classes with the largest possible margin.
- The margin width is $\dfrac{2}{\|w\|}$, so maximizing the margin is equivalent to minimizing $\|w\|$.
- The hard-margin linear SVM solves the primal problem $\min \frac{1}{2}\|w\|^2$ subject to $y_i(w^\top x_i + b) \geq 1$.
- Using Lagrange multipliers, we derive the dual problem in terms of α_i and dot products $x_i^\top x_j$.
- The optimal weight vector is $w = \sum_i \alpha_i^\star y_i x_i$, and only points with $\alpha_i^\star > 0$ (support vectors) influence the model.
- For a new point x, the prediction is $\hat{y} = \operatorname{sign}\big(w^\top x + b\big)$, or in dual form $\hat{y} = \operatorname{sign}\big(\sum_i \alpha_i^\star y_i K(x_i, x) + b\big)$.
- The kernel trick allows SVMs to learn non-linear decision boundaries by replacing dot products with kernel functions $K(x_i, x_j)$, without explicitly computing high-dimensional feature mappings.