Scientific Machine Learning 14
Support Vector Machines (SVM)

Donghyun Ko

January 4, 2026

What you will learn

Support Vector Machines (SVMs) pick the "safest" separating hyperplane by maximizing the margin—the distance to the nearest training points. Only those boundary-hugging points, the *support vectors*, influence the solution; the rest politely do not. In the hard-margin view, SVMs minimize $\frac{1}{2}\|w\|^2$ subject to $y_i(w^\top x_i + b) \geq 1$, yielding a sparse classifier. Reality is softer: slack variables and a single knob C trade margin width for a few well-chosen violations. Via Lagrangian duality and KKT conditions, the optimizer depends only on inner products, unlocking the kernel trick: replace $\langle x_i, x_j \rangle$ with $k(x_i, x_j)$ to draw linear planes in high (even infinite) dimensions while inducing nonlinear boundaries in input space. Linear, polynomial, RBF, and sigmoid kernels encode different notions of similarity—from finite interactions to universal smooth approximations. The result is a fast, robust, and tunable decision rule that generalizes well and separates spirals while pretending it's still just a straight line.

Contents

1 Introduction

Imagine drawing a line that splits two groups of points on a plane. There are countless ways to draw such a line, but which one feels the *most confident*? A Support Vector Machine (SVM) answers this question not by counting errors, but by measuring *distance*. It searches for the boundary that leaves the widest possible empty corridor between the two classes—a boundary that does not just separate, but separates with *margin*.

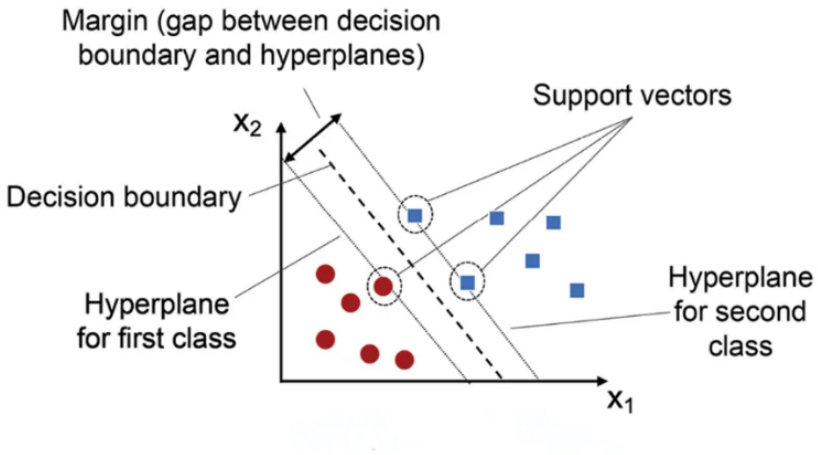


Illustration of SVM.

In this view, confidence is geometric. If a few points lean too close to the boundary, they become the "support vectors"—the critical witnesses that decide where the line must lie. Every other point fades into the background. These support vectors alone define the final classifier, as if the rest of the data simply agrees with their testimony. SVMs shine because they balance two competing instincts of learning: i) to fit the data faithfully and ii) to stay simple enough to generalize. A wide margin discourages overfitting, acting as an invisible regularizer that keeps the model humble. Yet life is rarely perfectly separable, so SVMs gracefully bend the rules by allowing a few points to step inside the margin in exchange for a smoother, more tolerant boundary. Then, it comes with the magic twist by mapping data into higher-dimensional spaces, SVMs can carve out complex, curved boundaries while still thinking in terms of straight lines. This trick—called the **kernel**—lets us handle spirals, rings, and any shape that refuses to be split by a flat line. Under the hood, it's still a linear model, but in a space so rich that linearity becomes expressive. We could understand this with an opposite lens of dimension reduction technics like PCA which are widely used in machine learning. They try to reduce the dimension by finding a fewer number of features that can explain the whole data to make model explainable, while kernel-based tricks try to expand the dimension so that we can find a hyperplane that can separate the entire dataset. Throughout this chapter, we will trace this journey from geometry to optimization and then to kernels: starting with the intuition of margins, then formulating the mathematical principles of "hard" and "soft" margins, unveiling how duality and kernels extend SVMs to the nonlinear world, and finally exploring their practical power in modern machine learning.

2 Fundamental Concepts

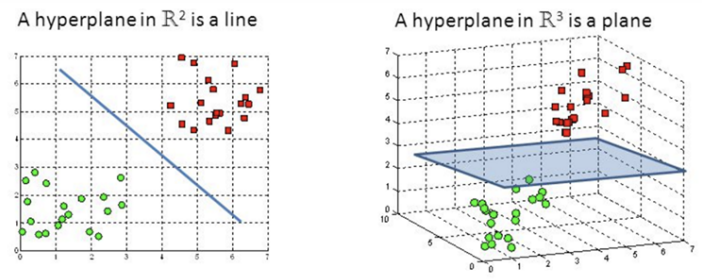
Support Vector Machine (SVM) is a **supervised machine learning algorithm** that can be used for both regression and classification tasks. However, it is most widely applied in classification problems, where the goal is to distinguish two or more categories based on feature patterns. The core objective of SVM is to find a *hyperplane* that separates different classes with the **largest possible margin** in a d-dimensional feature space ('d' is the number of features). In essence, SVM looks for the boundary that most confidently divides the data not merely one that separates, but one that does so with maximal distance from both sides.

Definition: Hyperplane and Decision Boundary

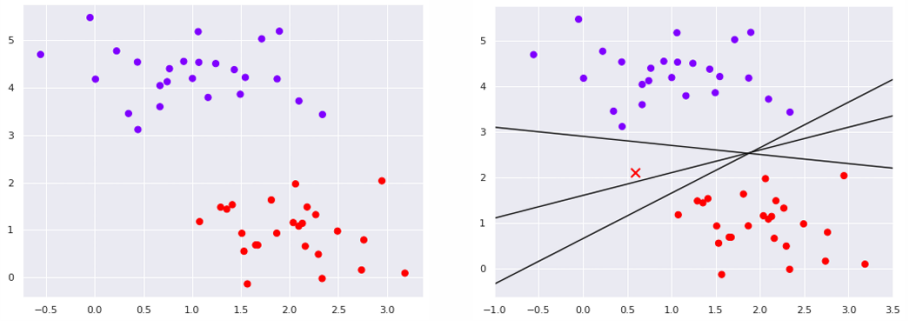
A *hyperplane* in $\mathbb{R}^d$ is a flat affine subspace defined as:

$$\mathcal{H} = \{x \mid w^\top x + b = 0\},$$

where w is the normal vector (perpendicular to the surface) and b is the bias term. This hyperplane acts as a **decision boundary**: points on one side belong to one class, while points on the other side belong to the opposite class. For $d = 1$, the decision boundary is a single point; for $d = 2$, it becomes a line; for $d = 3$, it is a plane; and for $d > 3$, it is called a **hyperplane**.

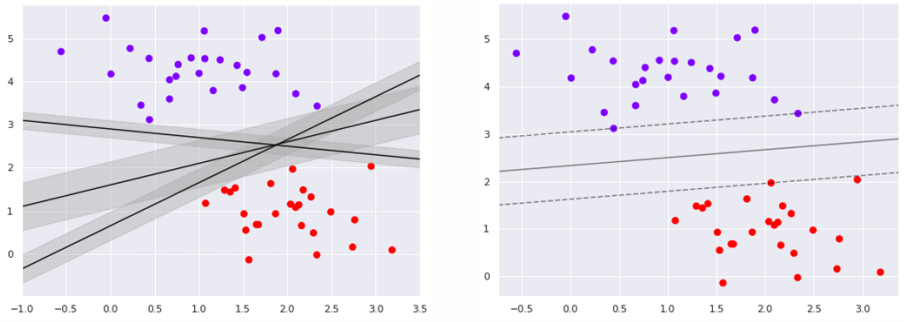


Consider a simple classification task in which two classes of points are well separated in space. A linear classifier would naturally attempt to draw a straight line that divides the two groups and then use this line to predict future points. However, there are infinitely many lines that can perfectly separate the data. Each line may lead to a different classification for a new test point ('x' in the figure below), depending on its position relative to the boundary. Thus, the key question arises: *which separating line (or hyperplane) should be chosen as the best model?*



Intuition Behind SVM

Rather than simply drawing a single boundary line between the classes, SVM imagines a **margin** of some width around that line — extending up to the nearest data points on each side. The optimal classifier is the one that *maximizes this margin*. Hence, SVM is often described as a **maximum margin classifier**. The figure below illustrates this concept: the right-hand plot shows the dividing line that maximizes the distance between the two classes. A few points from each class lie exactly on the margin — these are the critical points that determine the boundary and are known as the **support vectors**.



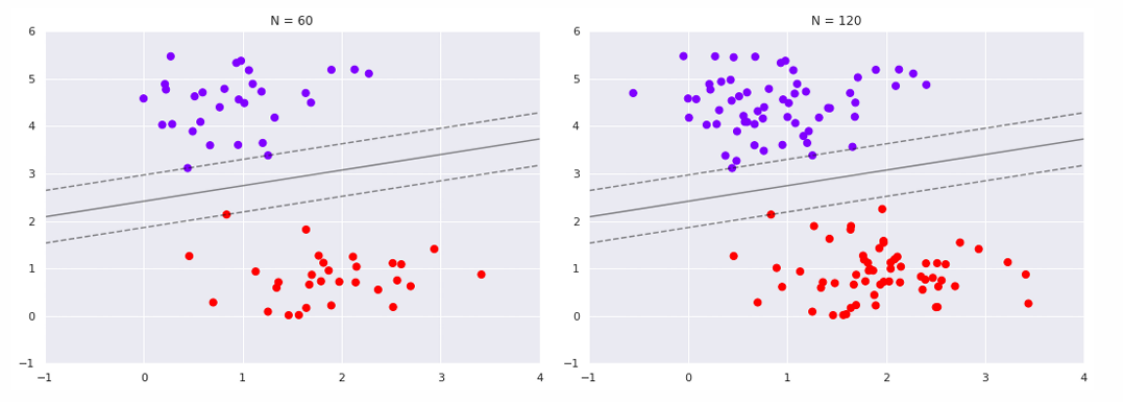
Definition: Margin and Support Vectors

- The **margin** is the perpendicular distance between two parallel lines (or hyperplanes) that pass through the closest data points of each class. These boundary lines represent the outer edges of the "street" separating the two groups.
- The data points that lie exactly on these boundaries are called **support vectors**. They are the most difficult to classify and have a direct effect on the optimal position and orientation of the decision surface.

If the margin between the classes is wide, the classifier is considered *good*: small variations in the data will not change its prediction. A small or narrow margin, on the other hand, indicates a fragile boundary that may easily misclassify new points. For this reason, the width of the margin is sometimes referred to informally as the **street width**.

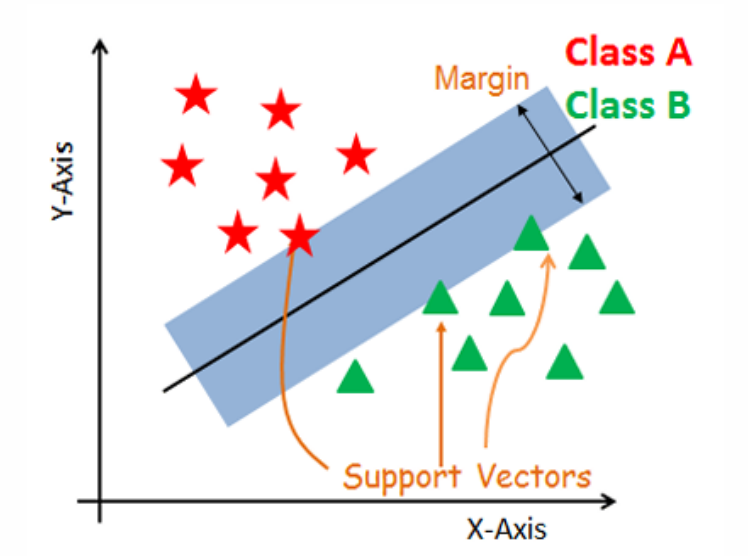
Why Only Support Vectors Matter

A key reason for SVM's success lies in its sparsity: *only the support vectors influence the solution.* Points far from the margin, which are already on the correct side of the boundary, do not affect the optimization. Technically, they contribute zero to the loss function used to fit the model. This means that even if we double the number of non-critical points (say from $N = 60$ to $N = 120$), the resulting SVM classifier remains the same — as long as the set of support vectors is unchanged. The insensitivity to the exact behavior of distant points is one of SVM's greatest strengths. Only a handful of boundary-defining points matter, giving SVM both robustness and computational elegance.



Summary

In the figure below, red stars and green triangles represent two **classes**. The blue shaded band between them is the **margin**, while the central black line is the **separating hyperplane**. The few data points touching the edges of this blue region are the **support vectors**. They are the only ones that determine the boundary — moving them slightly would shift the hyperplane, while moving other, distant points would have no effect.



3 Mathematical Theory

After understanding the geometric intuition behind Support Vector Machines, we now formalize the mathematics that enables SVM to find the **optimal separating hyperplane** — the one that maximizes the margin between two classes of data points.

The Equation of a Hyperplane

We begin by defining the equation of a general hyperplane:

$$w^\top x = 0$$

Here, $w = (w_1, w_2, \ldots, w_d)^\top$ is the **weight vector** normal (perpendicular) to the surface, and $x = (x_1, x_2, \ldots, x_d)^\top$ represents a point in the d-dimensional feature space.

Example (Linear case)

Consider a simple 2-D example where the hyperplane is a line:

$$x_2 = ax_1 + b$$

Rewriting this equation in homogeneous form gives

$$b + ax_1 - x_2 = 0 \quad \Longrightarrow \quad \begin{bmatrix} b \\ a \\ -1 \end{bmatrix}^\top \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = w^\top x = 0$$

Both forms represent the same geometric concept; the latter simply embeds the bias term b as an additional coordinate ($w_0 = -b$, $x_0 = 1$). It is more convenient, however, to express the hyperplane in the compact form of $w^\top x + b = 0$

Note that the vector w is orthogonal(normal) to the hyperplane — its direction determines the orientation of the decision boundary.

Proof that w is normal to the hyperplane

Consider the hyperplane $w_1x_1 + w_2x_2 + b = 0$. Take two points on this plane, $c_1 = (x_1^{(1)}, x_2^{(1)})$ and $c_2 = (x_1^{(2)}, x_2^{(2)})$, both satisfying

$$w_1x_1^{(1)} + w_2x_2^{(1)} + b = 0, \qquad w_1x_1^{(2)} + w_2x_2^{(2)} + b = 0$$

Subtracting the two equations (i.e., spanning the space with these two vectors) gives

$$w_1(x_1^{(1)} - x_1^{(2)}) + w_2(x_2^{(1)} - x_2^{(2)}) = 0 \quad \Longrightarrow \quad w^\top(c_1 - c_2) = 0$$

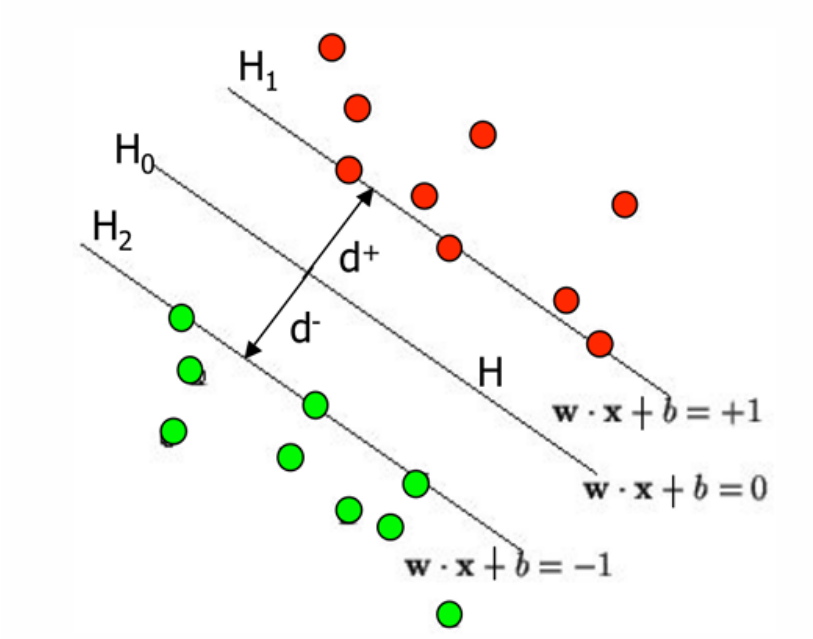
This means 'w' is orthogonal to any direction vector lying *on* the plane (like the line between c_1 and c_2). Hence, 'w' is the normal vector perpendicular to the hyperplane, and its direction sets the plane's orientation. □

Training Data and Classification Setup

Let our training dataset be

$$\mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathbb{R}^p,\ y_i \in \{+1, -1\}\}_{i=1}^{n}$$

Here x_i is the i-th training sample (a p-dimensional vector), y_i its label, and n the total number of samples. The goal of SVM is to determine (w, b) such that the hyperplane $w^\top x + b = 0$ best separates the positive and negative classes.



Two Parallel Boundaries and Constraints

To ensure separation, we define two parallel hyperplanes that bound the margin region:

$$H_1 : w^\top x + b = +1, \qquad H_2 : w^\top x + b = -1$$

No data points should fall between H_1 and H_2. The central hyperplane $H_0 : w^\top x + b = 0$ lies midway between them and serves as the decision boundary. Each sample (x_i, y_i) must therefore satisfy

$$\begin{cases} w^\top x_i + b \geq +1, & \text{if } y_i = +1 \\ w^\top x_i + b \leq -1, & \text{if } y_i = -1 \end{cases}$$

These two inequalities can be combined elegantly into a single constraint:

$$y_i(w^\top x_i + b) \geq 1, \quad \forall i = 1, \ldots, n$$

Geometrically, the condition $y_i(w^\top x_i + b) \geq 1$ guarantees that every point lies on the correct side of the margin boundaries. Equality ($y_i(w^\top x_i + b) = 1$) holds only for the *support vectors* that lie exactly on H_1 or H_2.

Explanation: unifying the two margin constraints

For a negative sample ($y_i = -1$), multiplying both sides of the inequality by y_i (that is, by -1) reverses the direction:

$$w^\top x_i + b \leq -1 \quad \Longrightarrow \quad y_i(w^\top x_i + b) \geq 1$$

For a positive sample ($y_i = +1$), the sign of the inequality is preserved:

$$w^\top x_i + b \geq +1 \quad \Longrightarrow \quad y_i(w^\top x_i + b) \geq 1$$

Thus, both class conditions can be unified into a single concise expression:

$$y_i(w^\top x_i + b) \geq 1, \qquad i = 1, \ldots, n$$

Margin and the Optimization Objective

The distance between these two hyperplanes defines the **margin width**, which SVM aims to maximize. Let d^+ denote the shortest distance from the decision boundary H_0 to the closest positive sample, and d^- the distance to the closest negative sample. If we balance the boundary symmetrically, these distances are equal ($d^+ = d^-$), and the total margin is:

$$m = d^+ + d^- = 2d^+ = 2d^-$$

Theorem: Distance from a Point to a Hyperplane

In a two-dimensional plane, the perpendicular distance from a point (x_0, y_0) to a line $Ax + By + C = 0$ is given by:

$$\boxed{d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}}$$

This result generalizes naturally to higher dimensions, where the denominator corresponds to the norm of the normal vector of the hyperplane.

This can be generalized to any d-dimensional hyperplane $w^\top x + b = 0$ as:

$$\text{dist}(x^*, H) = \frac{|w^\top x^* + b|}{\|w\|}$$

Applying this formula to the two margin boundaries $H_1 : w^\top x + b = +1$ and $H_2 : w^\top x + b = -1$, the perpendicular distances from H_0 are:

$$d^+ = d^- = \frac{1}{\|w\|},$$

so that the margin width ('m') becomes

$$m = \frac{2}{\|w\|}$$

Thus, maximizing the margin m is equivalent to minimizing $\|w\|$. To sum up, training a SVM model now becomes a constrained (convex) optimization problem such that:

$$\boxed{\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i\Big(w^\top x_i + b\Big) \;\geq\; 1, \qquad i = 1, \ldots, n \end{aligned}}$$

This objective maximizes the geometric margin $m = 2/\|w\|$ while enforcing every training point to lie on or outside its class-specific margin boundary.

Note

Geometric Insight. The weight vector 'w' not only determines the direction of the hyperplane but also controls how "tight" the classifier is. A larger $\|w\|$ corresponds to a narrower street (smaller margin), while a smaller $\|w\|$ gives a wider, more confident boundary.

Hard-Margin Primal Optimization Problem

We can now formalize the goal of finding the widest margin as a constrained convex optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(w^\top x_i + b) \geq 1, \quad i = 1, \ldots, n \end{aligned} \tag{P}$$

The objective $\frac{1}{2}\|w\|^2$ penalizes large weights and hence implicitly maximizes the margin $2/\|w\|$. The constraints ensure that all samples are correctly classified and the margin boundaries touch the nearest points.

Lagrangian Dual Formulation and KKT Conditions

To solve this constrained problem efficiently, we employ the method of Lagrange multipliers. Introduce $\alpha_i \geq 0$ for each constraint and define the Lagrangian:

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

The stationarity conditions require:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i, \qquad \text{and} \qquad \frac{\partial \mathcal{L}}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Substituting these into $\mathcal{L}$ eliminates w and b, yielding the dual objective:

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

Hence, the dual problem is:

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0 \end{aligned} \tag{D}$$

KKT Conditions and Support Vectors

At optimality, the Karush–Kuhn–Tucker (KKT) conditions hold:

$$\alpha_i \Big(y_i (w^\top x_i + b) - 1 \Big) = 0$$

Thus, $\alpha_i > 0$ only for points satisfying $y_i(w^\top x_i + b) = 1$, i.e., the *support vectors*. These are the only data points that determine the final classifier.

The resulting decision function is:

$$f(x) = \operatorname{sign}\left(\sum_{i \in \mathcal{S}} \alpha_i y_i \langle x_i, x \rangle + b \right),$$

where $\mathcal{S}$ is the set of support vectors and b can be computed as

$$b = y_s - \sum_{i \in \mathcal{S}} \alpha_i y_i \langle x_i, x_s \rangle, \quad \text{for any support vector } x_s.$$

The dual depends only on inner products $\langle x_i, x_j \rangle$, which opens the door to nonlinear mappings via **kernels** which is a topic we explore next.

Summary

Training a hard-margin SVM reduces to the following elegant principle:

$$\boxed{\text{Find } (w, b) \text{ that minimize } \|w\| \text{ subject to } y_i(w^\top x_i + b) \geq 1.}$$

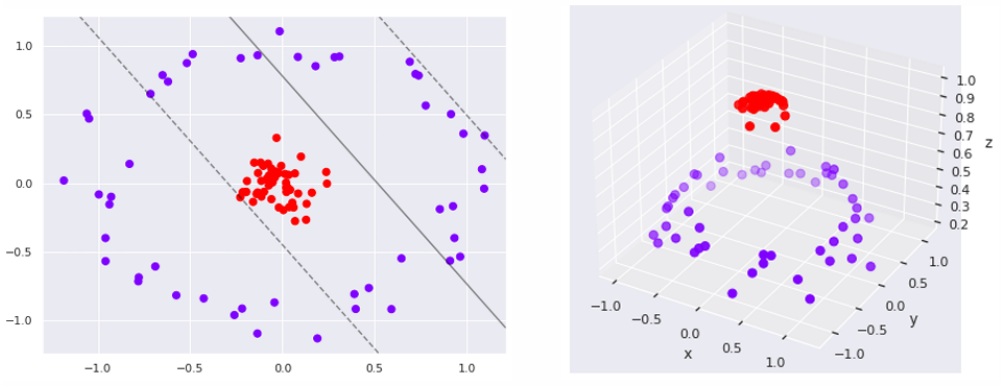
This connects geometry (maximum margin) with optimization (minimum norm). To fully solve this, one must understand **constrained optimization, duality, and Lagrange multipliers**—the mathematical foundations upon which SVM is built.

Further Reading

1. A. Kowalczyk, *"SVM Tutorial: Math behind SVM."* svm-tutorial.com
2. J. Friedman, T. Hastie, R. Tibshirani, *The Elements of Statistical Learning*, Ch12.
3. A. Kumar, *"Math behind Support Vector Machine,"* Medium, 2020. ankitnitjsr13.medium.com
4. Analytics Vidhya, *"The Mathematics Behind SVM."* analyticsvidhya.com

4 The Kernel Trick

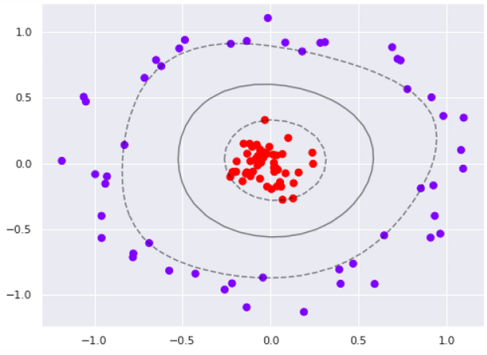
Sometimes, no matter how we rotate or shift the separating line, no *linear* boundary can perfectly separate the data in the original input space. Classic examples include ring-shaped or spiral datasets where the two classes are intertwined. A natural idea is to **transform the data into a higher-dimensional space** where a linear separation *becomes possible*. SVM becomes truly powerful when combined with the idea of a **kernel**, which allows it to handle *nonlinear input spaces*. Some datasets are simply not linearly separable in their original coordinates. No straight line (or hyperplane) in the input space can divide the two classes cleanly as illustrated in the left figure below.



To separate these nonlinear patterns, we can introduce a new feature to expand the dimension. For example, define

$$z = e^{-(x^2+y^2)}$$

Now each data point (x, y) becomes a 3D point (x, y, z). In this new feature space, the data that was tangled in 2D can be separated easily by a 2D hyperplane. The right-hand figure in the above shows this lifted 3D space, where the two clusters (originally overlapping rings) become linearly separable when viewed along the z-axis. Once the data is lifted into a richer feature space, the SVM learns a *linear hyperplane* there. When projected back to the original input space, this linear boundary manifests as a *nonlinear decision boundary*. Hence, SVMs can construct curved boundaries simply by applying linear separation in a transformed space.



Using such *kernelized SVMs*, we can learn nonlinear decision boundaries efficiently, without ever explicitly constructing the higher-dimensional features.

Definition: The Kernel and the Kernel Trick

A **kernel function** is a measure of similarity between two samples, defined as

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}},$$

where $\phi : \mathbb{R}^p \to \mathcal{H}$ is a (possibly implicit like a manifold) feature mapping into a higher-dimensional Hilbert space $\mathcal{H}$. In practice, we never compute $\phi(x)$ directly. Instead, we compute the kernel value $k(x_i, x_j)$ which is the inner product in the transformed space, without ever constructing the features themselves.

Key idea (the "trick"):

- Replace every dot product $\langle x_i, x_j \rangle$ in the SVM optimization with $k(x_i, x_j)$.
- The SVM then finds a *linear separator* in the feature space $\mathcal{H}$, which corresponds to a *nonlinear boundary* in the original input space.

Intuitive meaning:

- Kernels let SVMs act as if they had mapped data into a rich, high-dimensional space, even when the transformation is infinite-dimensional.
- The computation stays entirely in the input space, meaning that there is no need to explicitly generate new features.

From the SVM primal problem to the kernelized dual. Training an SVM ultimately means finding the hyperplane that maximizes the geometric margin between two classes. The *primal optimization problem* for the hard-margin case is:

$$\min_{w,b} \ \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i \left(w^\top x_i + b\right) \geq 1, \quad i = 1, \ldots, n \tag{$\mathrm{P}_{\mathrm{hard}}$}$$

Introducing Lagrange multipliers $\alpha_i \geq 0$ for each constraint gives the Lagrangian:

$$\mathcal{L}(w, b, \alpha) = \tfrac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \Big(y_i (w^\top x_i + b) - 1 \Big)$$

Setting the derivatives to zero yields the stationary conditions:

$$\boxed{\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i}, \quad \text{and} \quad \boxed{\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0}$$

Thus, the optimal weight vector is a linear combination of the training samples, and the bias term b is determined by the support vectors that touch the margin. Substituting w back into $\mathcal{L}$ eliminates w and b, producing the *dual problem*:

$$\max_{\alpha} \ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \quad \text{s.t.} \quad \alpha_i \geq 0, \ \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{$\mathrm{D}_{\mathrm{hard}}$}$$

The resulting decision function is:

$$f(x) = \operatorname{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i \langle x_i, x\rangle + b\Big), \qquad \alpha_i > 0 \;\Rightarrow\; x_i \text{ is a support vector.}$$

For the soft-margin SVM (with slack variables ξ_i and penalty C), the primal becomes:

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

The corresponding dual becomes:

$$\max_{\alpha} \; \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle, \quad \text{s.t.} \quad 0 \leq \alpha_i \leq C,\; \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{D_{soft}}$$

Kernel Tricks!

In both dual formulations, note that all data dependencies appear *only* through the inner products $\langle x_i, x_j\rangle$ and $\langle x_i, x\rangle$. This means that we can implicitly map each input x into a higher-dimensional feature space $\mathcal{H}$ using some transformation $\phi : \mathbb{R}^p \to \mathcal{H}$, and replace every dot product by

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle_{\mathcal{H}}.$$

We never need to compute $\phi(x)$ explicitly, instead we only need $k(x_i, x_j)$. This substitution is what we call the **kernel trick**.

Kernelized dual and prediction function. Let $\phi : \mathbb{R}^p \to \mathcal{H}$ be a feature map into a higher-dimensional (possibly infinite) Hilbert space. If we could compute all transformed vectors $\phi(x_i)$, then the dual SVM problem (D) would involve their inner products:

$$\max_{\alpha} \; \sum_{i} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \langle \phi(x_i), \phi(x_j)\rangle_{\mathcal{H}}, \quad \text{s.t.} \;\; \sum_{i} \alpha_i y_i = 0,\; \alpha_i \geq 0$$

However, the key insight is that we do not need to compute $\phi(x)$ explicitly, instead we only need the inner products $\langle \phi(x_i), \phi(x_j)\rangle$. If we define a function

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle_{\mathcal{H}},$$

then the entire optimization can be expressed in terms of 'k' alone, and this is the dual problem (D) of the primal SVM problem (P). Therefore, in a more general way, after applying the kernel trick, the dual optimization becomes:

$$\max_{\alpha} \; \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j\, k(x_i, x_j), \quad \text{s.t.} \quad \begin{cases} \alpha_i \geq 0 & \text{(hard margin)}, \\ 0 \leq \alpha_i \leq C & \text{(soft margin)}, \end{cases} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

The resulting prediction rule is:

$$f(x) = \operatorname{sign}\left(\sum_{i\in\mathcal{S}} \alpha_i y_i\, k(x_i, x) + b\right), \quad \mathcal{S} = \{i : \alpha_i > 0\} \text{ are the support vectors.}$$

Interpretation. After kernelization, the weight vector becomes

$$w = \sum_{i=1}^{n} \alpha_i y_i \, \phi(x_i),$$

so the classifier is linear in the feature space $\mathcal{H}$, yet nonlinear in the original input space. The SVM therefore learns a linear separator in a transformed space, which corresponds to a *nonlinear decision boundary* in the input domain.

Theorem (Mercer's condition)

A symmetric function $k(x, x')$ is a valid kernel if and only if, for any finite set $\{x_1, \ldots, x_n\}$, the Gram matrix

$$K = [k(x_i, x_j)]_{i,j}$$

is **positive semidefinite (PSD)**. In that case, there exists a feature map ϕ and a Hilbert space $\mathcal{H}$ such that

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}.$$

Meaning: Kernels serve as stand-ins for an explicit feature lift. They let us perform computations in high (even infinite) dimensions, while staying entirely in the original input space.

Popular Kernels

Each kernel function implicitly defines a geometry in feature space, determining how similarity between two samples is measured. Different kernels thus correspond to different notions of smoothness, curvature, and flexibility in the learned decision boundary.

(1) Linear Kernel. The most elementary kernel is the **linear kernel**,

$$k(x, x') = \langle x, x' \rangle = \sum_{j=1}^{p} x_j x'_j.$$

It represents the ordinary dot product in the input space itself, meaning no nonlinear transformation is applied. The resulting SVM searches directly for a hyperplane in the original coordinate system. Because of its simplicity, this kernel is well suited for linearly separable data or high-dimensional problems where $p \gg n$.

Interpretation

The linear kernel measures similarity through pure alignment: two samples have a high kernel value when they point in similar directions. It is most effective when features already capture meaningful linear relations.

(2) Polynomial Kernel. To model interactions among features, we can extend the linear kernel into its polynomial form:

$$k(x, x') = (\gamma\langle x, x'\rangle + r)^d,$$

where γ controls scaling, r is a shifting constant, and d denotes the degree of the polynomial. When $d = 1$, this reduces exactly to the linear kernel. A polynomial kernel implicitly augments the feature space with higher-order interaction terms such as x_i^2 or $x_i x_j$. And it corresponds to an inner product in a finite-dimensional feature space whose basis functions are all monomials of total degree $\leq d$. Thus, the polynomial kernel induces an explicit finite expansion in contrast to the RBF kernel, whose expansion is infinite. For example, with one-dimensional input, the polynomial kernel of degree two expands to new quadratic features even though they are never explicitly computed.

Example (1D feature expansion)

Consider the kernel

$$k(x, x') = (xx' + \tfrac{1}{2})^2.$$

Expanding gives

$$(xx' + \tfrac{1}{2})^2 = x^2x'^2 + xx' + \tfrac{1}{4} = [x^2,\, x,\, \tfrac{1}{2}] \cdot [x'^2,\, x',\, \tfrac{1}{2}]^\top.$$

Hence, the implicit feature map is

$$\phi(x) = [x^2,\, x,\, \tfrac{1}{2}]^\top,$$

which reveals that the polynomial kernel automatically introduces the higher-order term x^2. The constant $\frac{1}{2}$ does not count as a new feature, since it is shared by all samples. A slightly different example,

$$k(x, x') = (xx' + 1)^2 = x^2x'^2 + 2xx' + 1 = [x^2,\, \sqrt{2}x,\, 1] \cdot [x'^2,\, \sqrt{2}x',\, 1]^\top,$$

shows that the new implicit features are $\sqrt{2}x$ and x^2. In practice, the SVM never construct these terms explicitly, instead it only evaluates the kernel to infer their dot-product structure.

(3) Radial Basis Function (RBF) or Gaussian Kernel. Perhaps the most widely used kernel is the **RBF kernel**, defined as

$$k(x, x') = \exp(-\gamma\|x - x'\|^2),$$

where $\gamma > 0$ controls how far the influence of a single training sample extends. A small γ produces a wide, smooth similarity region around each point (underfitting), whereas a very large γ yields sharp local effects (overfitting).

Guidelines for choosing kernels

Use a **linear kernel** when the number of features is larger than the number of observations. Use an **RBF kernel** when the dataset has far more samples than features.

To understand why the RBF kernel maps data into an *infinite-dimensional* space, consider a one-dimensional input with $x = a$ and $x' = b$, and let $\gamma = \frac{1}{2}$ for simplicity. Then,

$$k(a,b) = e^{-\frac{1}{2}(a-b)^2} = e^{-\frac{1}{2}(a^2+b^2-2ab)} = e^{-\frac{1}{2}(a^2+b^2)}\, e^{ab}.$$

Using the Taylor series expansion of the exponential,

$$e^{ab} = 1 + (ab) + \frac{(ab)^2}{2!} + \frac{(ab)^3}{3!} + \cdots,$$

we can rewrite the kernel as

$$k(a,b) = e^{-\frac{1}{2}(a^2+b^2)} \sum_{m=0}^{\infty} \frac{(ab)^m}{m!} = \phi(a)^\top \phi(b),$$

where

$$\phi(x) = e^{-\frac{1}{2}x^2}\left(1,\ x,\ \tfrac{x^2}{\sqrt{2!}},\ \tfrac{x^3}{\sqrt{3!}},\ \ldots\right).$$

Theorem: RBF Kernel Maps into Infinite Dimensions

The Gaussian kernel corresponds to the inner product

$$k(x,x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

in an infinite-dimensional Hilbert space $\mathcal{H}$, whose basis consists of all monomials of every degree scaled by $1/\sqrt{m!}$ and weighted by $e^{-\|x\|^2/2}$. Hence, RBF kernels capture infinitely many nonlinear interactions while remaining computationally tractable through the kernel trick.

Comparison: Polynomial vs. RBF Kernel

Polynomial kernels yield a finite feature expansion whose degree d is user-specified. RBF kernels, in contrast, correspond to an infinite expansion with exponentially decaying coefficients. Thus, RBF kernels offer universal approximation power—the ability to fit any smooth decision boundary given sufficient data and appropriate regularization.

(4) Sigmoid Kernel. The **sigmoid kernel** takes the form

$$k(x,x') = \tanh(\gamma\langle x, x'\rangle + r),$$

where tanh acts as the activation function in a neural network. Indeed, an SVM equipped with a sigmoid kernel is mathematically equivalent to a two-layer perceptron with a single hidden layer of hyperbolic-tangent neurons. While it provides smooth nonlinear separation, it is not guaranteed to be positive semidefinite for all parameter combinations (γ, r), so practical use requires careful tuning.

Summary and Remarks

Kernels transform the geometry of the input space without explicitly computing new coordinates. Because the dual SVM optimization depends only on pairwise inner products, we can substitute any valid positive-semidefinite kernel to implicitly work in high dimensions. Linear, polynomial, RBF, and sigmoid kernels each define a unique notion of similarity—ranging from rigid linear boundaries to highly flexible nonlinear manifolds.

Key Takeaways

- The **linear kernel** operates in the original feature space and assumes approximate linear separability.
- The **polynomial kernel** introduces finite-degree interactions, enriching the representational power.
- The **RBF kernel** implicitly maps inputs into infinite dimensions, yielding smooth and universal nonlinear boundaries.
- The **sigmoid kernel** connects SVMs with neural networks, bridging statistical learning and deep representations.

5 Soft Margins

In practice, data are rarely perfectly separable. Real-world dataset often contain noise, mislabeled examples, or inherent overlap between classes. For such cases, enforcing a perfectly clean boundary would lead to a model that overfits the training data, capturing outliers instead of the underlying structure.

Motivation: From Hard to Soft Margins

When the data are linearly separable and no misclassification is acceptable, the SVM seeks a **hard margin**, ensuring all points satisfy

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

However, this rigid formulation becomes infeasible if classes overlap. To relax it, we introduce a *fudge factor* that allows certain violations when they improve the overall fit. This leads to the concept of a **soft margin**.

Idea: Controlled Margin Violations

Soft-margin SVM allows a few points to lie inside the margin or even on the wrong side of the boundary if that trade-off improves the model's generalization. The flexibility is controlled by a regularization constant $C > 0$, which penalizes excessive violations.

Primal Problem with Slack Variables

Formally, we introduce nonnegative slack variables $\xi_i \geq 0$ representing the amount by which each constraint is violated:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i, \tag{1}$$

$$\text{s.t.} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0. \tag{2}$$

The first term in (1) keeps the margin wide, while the second penalizes constraint violations. Parameter C therefore determines the hardness of the margin: a large C discourages violations (approaching a hard margin), while a small C tolerates more slack (a smoother decision boundary).

Mathematical Interpretation

If $\xi_i = 0$, the ith sample lies correctly outside the margin. If $0 < \xi_i < 1$, it lies inside the margin but on the correct side. If $\xi_i > 1$, it is misclassified. Thus, $\sum_i \xi_i$ measures the total degree of violation across all samples.

Dual Formulation and KKT Conditions

The Lagrangian of the primal problem is:

$$\mathcal{L}(w, b, \xi, \alpha, \mu) = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i \big[y_i(w^\top x_i + b) - 1 + \xi_i\big] - \sum_i \mu_i \xi_i,$$

where $\alpha_i \geq 0$ and $\mu_i \geq 0$ are Lagrange multipliers. Setting the partial derivatives to 0 yields:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_i \alpha_i y_i x_i, \quad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_i \alpha_i y_i = 0, \quad \frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \mu_i = 0.$$

Since $\mu_i \geq 0$, it follows that $0 \leq \alpha_i \leq C$. Eliminating w, b, ξ gives the **dual problem**:

$$\max_{0 \leq \alpha_i \leq C} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \, k(x_i, x_j), \quad \text{s.t.} \quad \sum_i \alpha_i y_i = 0.$$

Theorem: Relationship between Hard and Soft Margins

The hard-margin SVM is a special case of the soft-margin formulation with $C \to \infty$. In this limit, all violations become infinitely costly, forcing every data point to lie outside the margin.

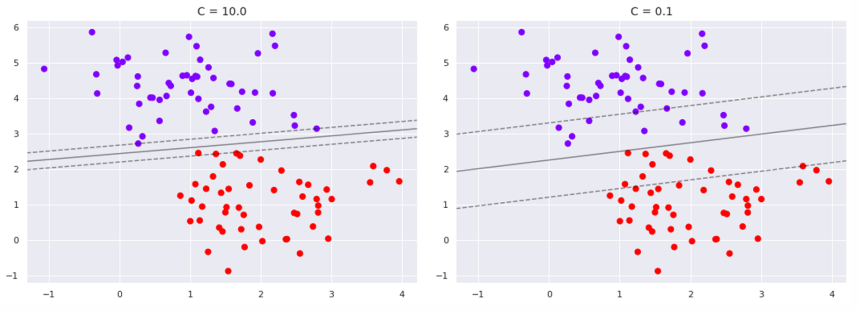
The decision function for any kernelized soft-margin SVM remains

$$f(x) = \text{sign}\left(\sum_{i \in \mathcal{S}} \alpha_i y_i \, k(x_i, x) + b\right),$$

where $\mathcal{S} = \{i : \alpha_i > 0\}$ are the support vectors.

Interpretation of C

For very large C, the SVM enforces a hard margin, classifying all training samples correctly but risking overfitting. For small C, the margin becomes softer and smoother, tolerating some misclassifications and thus generalizing better.



6 Discussions

Support Vector Machines combine geometric intuition, convex optimization, and kernel theory into a powerful and elegant framework. Yet, as with any method, their success depends on appropriate regularization and kernel selection.

Advantages of SVM

- Once trained, prediction is extremely fast and memory efficient, as only a few support vectors define the boundary.
- SVMs are highly effective in high-dimensional spaces, maintaining robustness even when $p \gg n$.
- Their compact representation and sparse dependence make them well-suited for large-feature problems such as text and image classification.
- They are versatile: various kernel functions (linear, polynomial, RBF, sigmoid, or even custom kernels) adapt SVMs to diverse data geometries.

Disadvantages of SVM

- Training can be computationally expensive for large datasets due to quadratic programming complexity.
- Performance depends strongly on appropriate tuning of the *softening parameter* C and kernel hyperparameters.
- Overlapping or noisy classes can degrade performance, requiring careful regularization.
- When $p \gg n$, improper kernel choice can cause overfitting; regularization becomes crucial.
- SVM outputs are not probabilistic by default and require post-hoc calibration (e.g., Platt scaling).

Key Takeaways

One-sentence summary. SVM finds the boundary *furthest from danger* by focusing on only the few points that matter—the support vectors—while balancing margin width and classification accuracy through the parameter C.

This article is written for study purposes and independently elaborates on NE-795 (Scientific Machine Learning) lecture concepts by Prof. Xu Wu, NC State University.