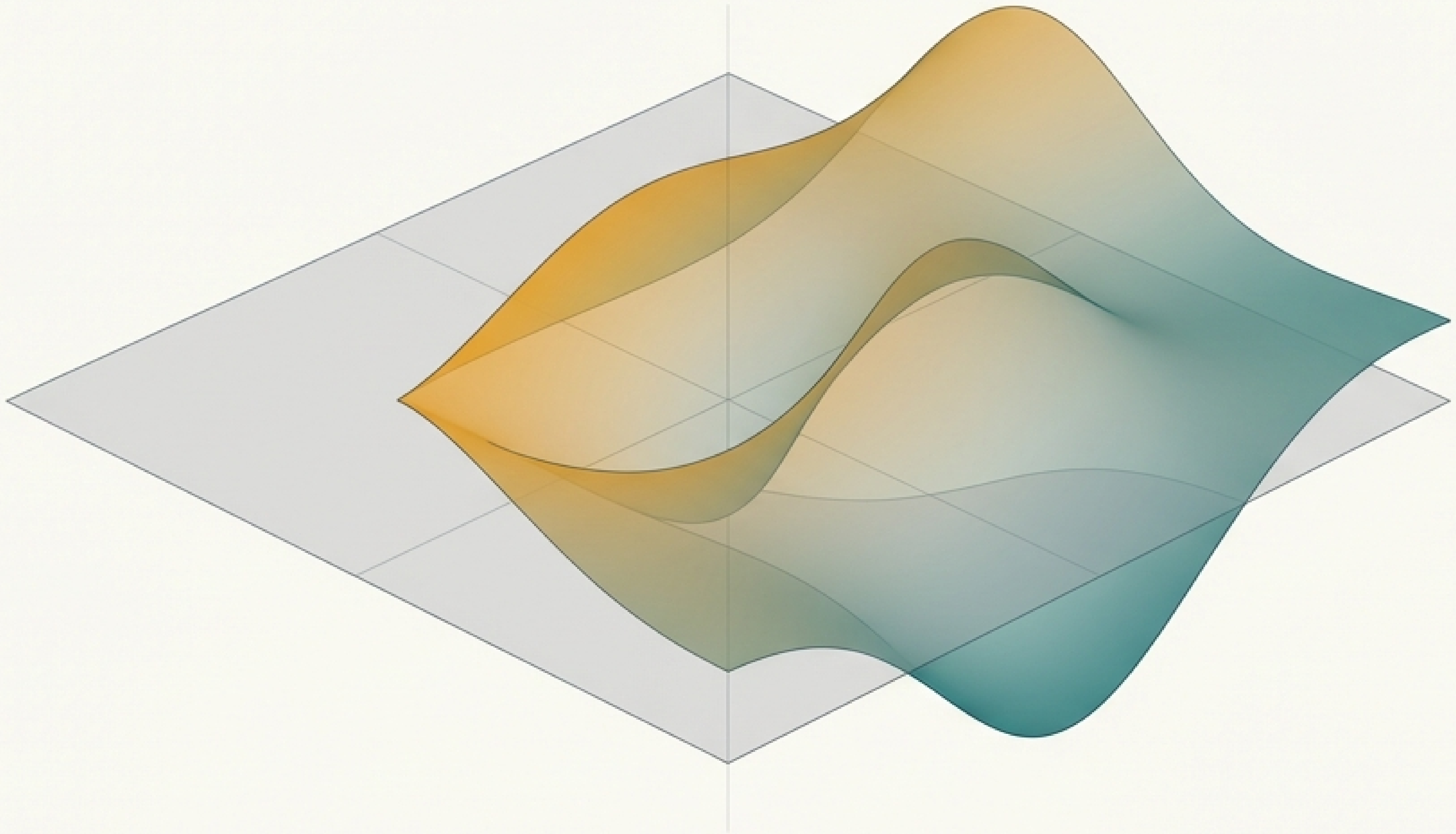


SVM & Kernels: A Chapter 9 Review

Mastering the Margin: From Linear Separation to Non-Linear Classification

NotebookLM

The Quest for the Optimal Hyperplane

Shared Goal:

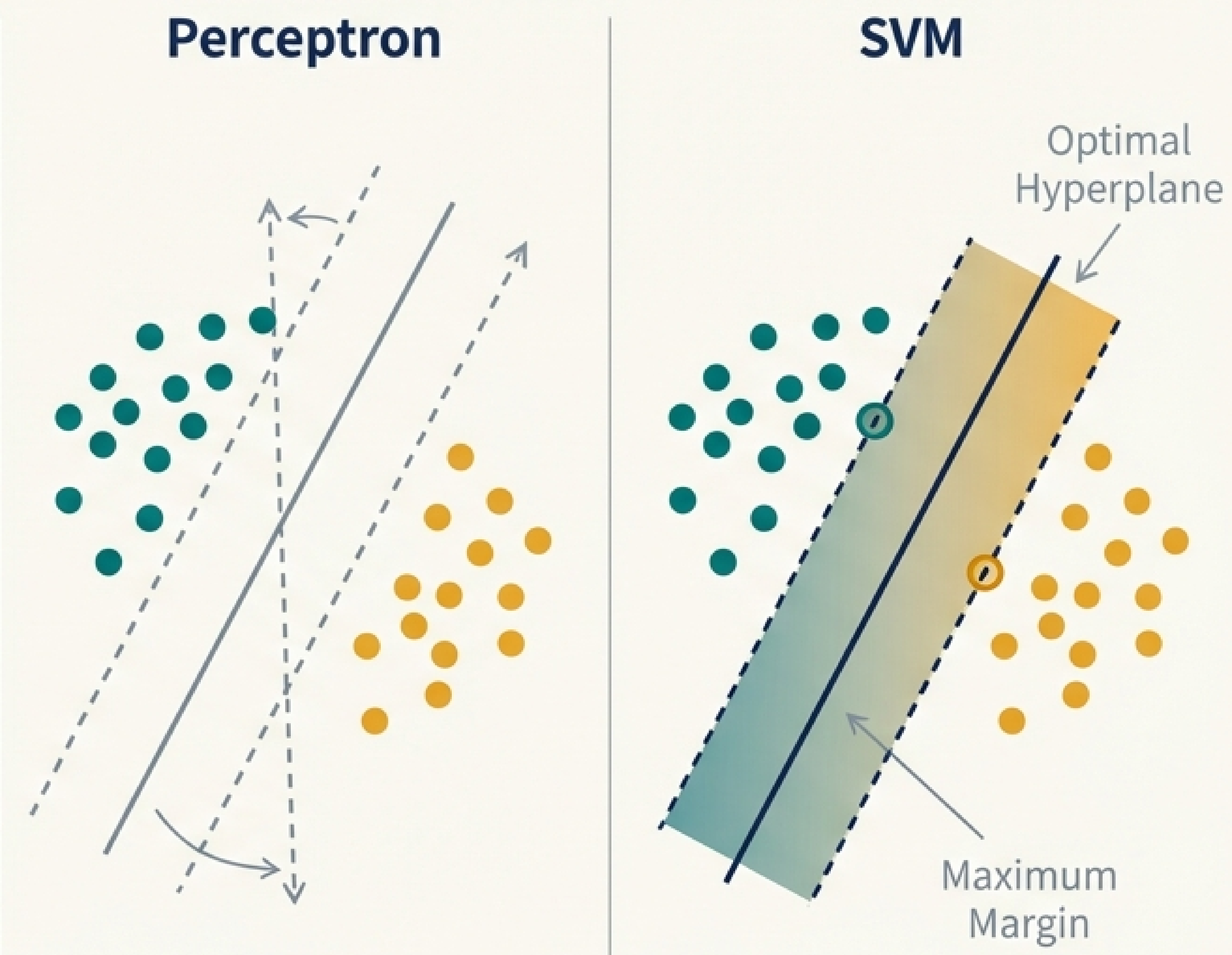
Both Perceptron and SVM are supervised methods that find a hyperplane ($\boldsymbol{w}^T\boldsymbol{x} + b = 0$) to separate two classes.

The Key Difference (from Exercise 6):

- **Perceptron:** Finds any separating hyperplane. Its solution can be arbitrary and depends on the starting point and data order.
- **SVM:** Seeks the unique hyperplane that maximizes the margin—the distance to the nearest data points of any class.

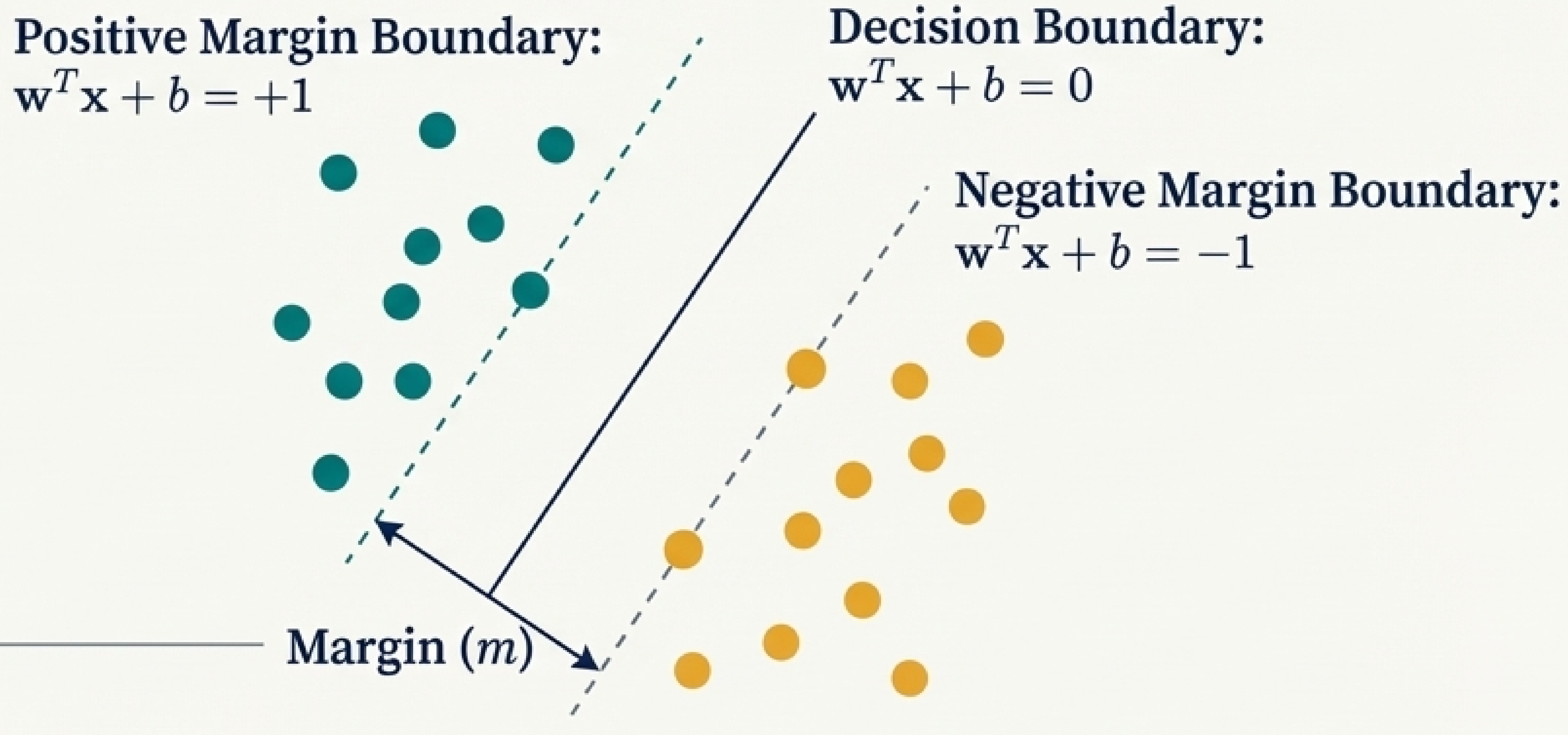
Intuition:

Maximizing the margin creates a more confident and robust decision boundary, leading to better generalization performance on unseen data.



The Geometry of Confidence

The margin is defined by three parallel hyperplanes. The scaling parameter `$s$` is set to 1 without loss of generality, which simplifies the formulation.



The Margin (m`): The perpendicular distance between the two outer hyperplanes is $m = \frac{2}{||\mathbf{w}||} = \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$.

The Goal: To maximize the margin m, we must minimize $||\mathbf{w}||$. For mathematical convenience, we minimize $\frac{1}{2}w^T\mathbf{w}$.

NotebookLM

Formalizing the Objective: The Primal Problem

Objective: Find the parameters w and b that:

$$\text{minimize}\ (w, b)\ \left\{\frac{1}{2}w^T w\right\}$$

Subject to Constraints: Every data point must be on or outside the correct side of its margin boundary.

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

Where y_i is the class label (+1 or -1).

This is a constrained convex optimization problem (specifically, a quadratic programming problem). Its convexity guarantees that there is a unique, globally optimal solution.

A New Perspective: The Dual Problem

The Lagrangian

We introduce the Lagrangian to incorporate the constraints into the objective:

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum \alpha_i \left[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1\right]$$

Minimization Results

By minimizing L with respect to w and b (setting derivatives to zero), we find two key relations:

$$\boldsymbol{w} = \sum \alpha_i y_i \boldsymbol{x}_i$$

$$\sum \alpha_i y_i = 0$$

The weight vector is a linear combination of data points.

The Dual Formulation

Substituting these back into the Lagrangian yields the dual problem (as derived in **Exercise 1**):

$$\text{maximize}\,(\boldsymbol{\alpha})\left\{\sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j\right\}$$

Subject to: $\sum \alpha_i y_i = 0$ and $\alpha_i \geq 0$.

Support Vectors: The Data Points That Matter

The solution to the dual problem provides the Lagrange multipliers, α_i.

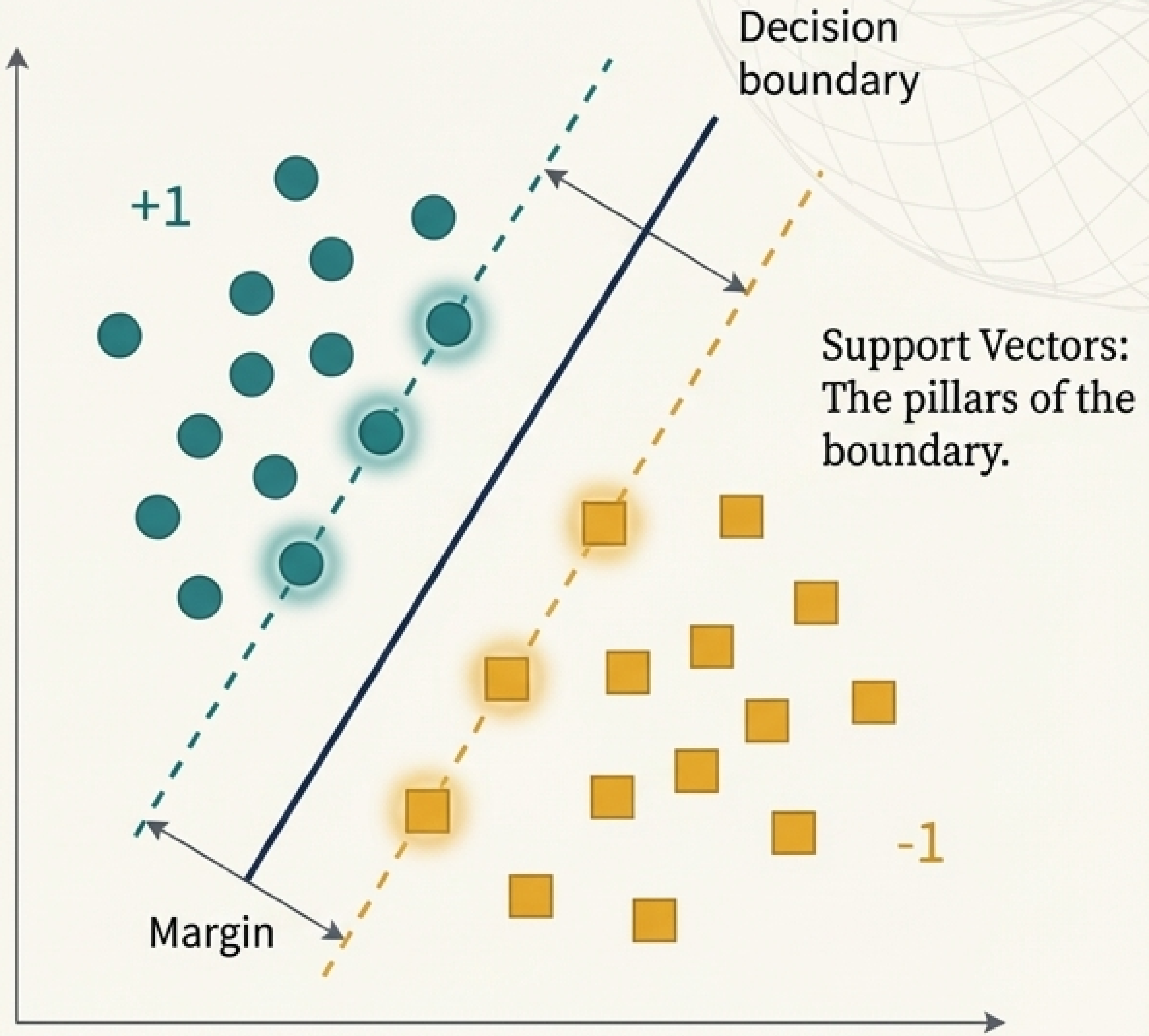
The Karush-Kuhn-Tucker (KKT) conditions include a property known as **Complementary Slackness**:

$$\boxed{\alpha_i \left[y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right] = 0 \quad \text{for each data point } i.}$$

The Implication: For α_i to be non-zero ($\alpha_i > 0$), the term in the brackets must be zero. This means the point $\mathbf{x}_i$ lies exactly on the margin boundary: $y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$.

These critical points with $\alpha_i > 0$ are called **Support Vectors.**

The decision boundary is determined *only* by these support vectors. As argued in **Exercise 9**, removing any non-support vector would not change the final hyperplane.



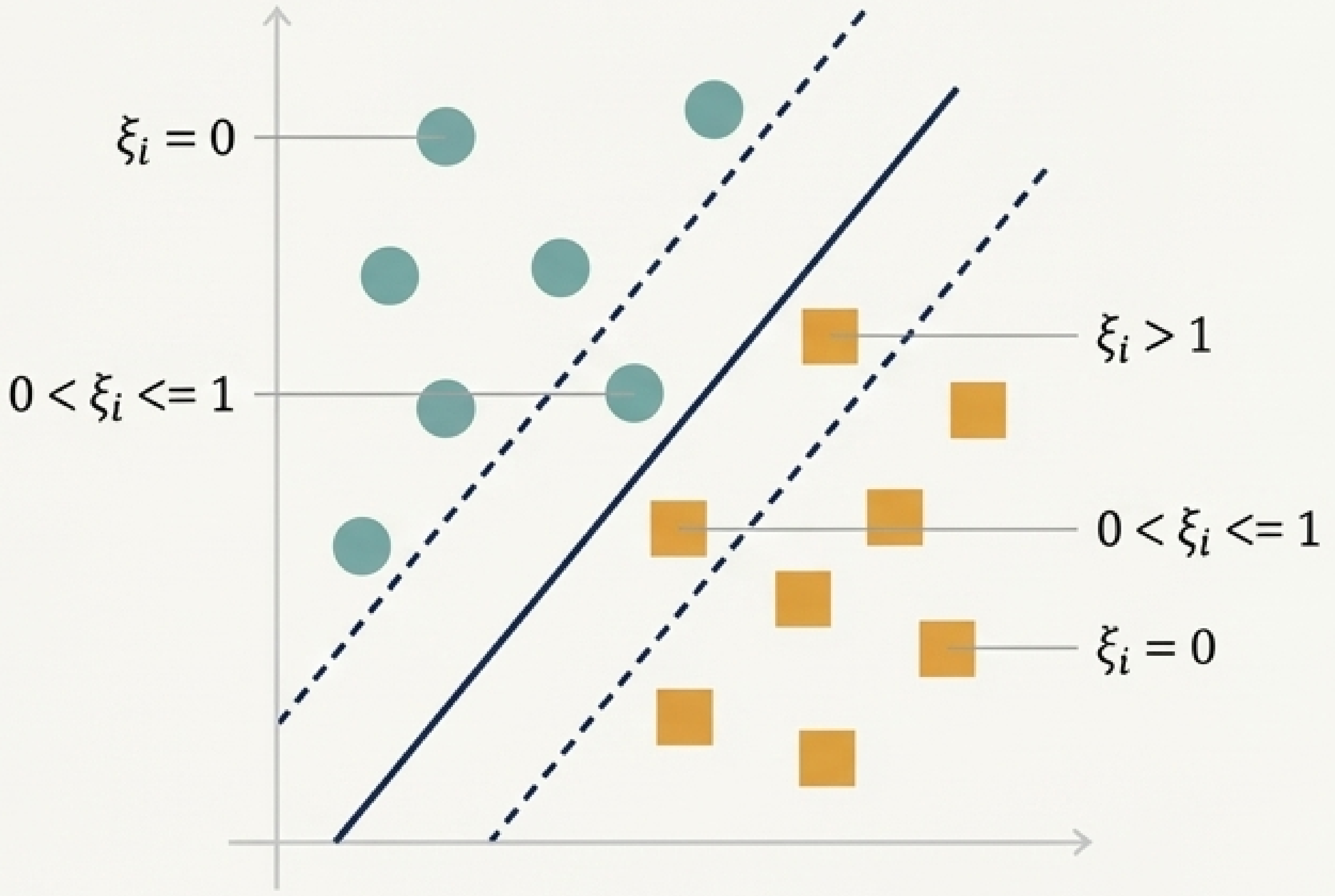
Embracing Imperfection: The Soft-Margin SVM

The Problem

In real-world data, perfect linear separation is rare. The hard-margin constraints $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$ can become impossible to satisfy.

The Solution

We introduce a "slack" variable $\xi_i >= 0$ for each data point to allow for some error. The constraint is relaxed to:
$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) >= 1 - \xi_i$.



Interpreting Slack (ξ_i)

- $\xi_i = 0$: Point is correctly classified and outside or on the margin.
- $0 < \xi_i <= 1$: Point is correctly classified but is inside the margin.
- $\xi_i > 1$: Point is misclassified.

The Optimization Trade-off: Margin vs. Misclassification

- **New Primal Problem:** We add a penalty term for the total slack to the objective function.

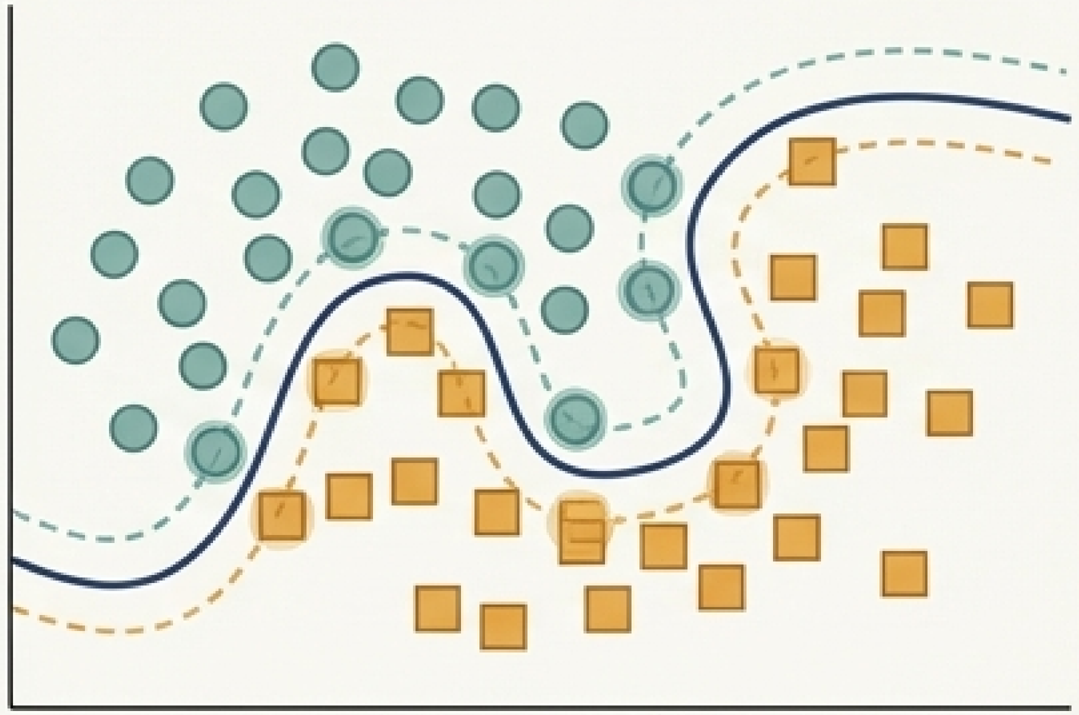
$$\underset{(\mathbf{w}, \mathbf{b}, \xi)}{\text{minimize}} \left\{ \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \right\}$$

  The hyperparameter C is a penalty for constraint violations.

- **The Dual Problem:** The only modification is a new upper bound on the Lagrange multipliers, creating a “box constraint”.

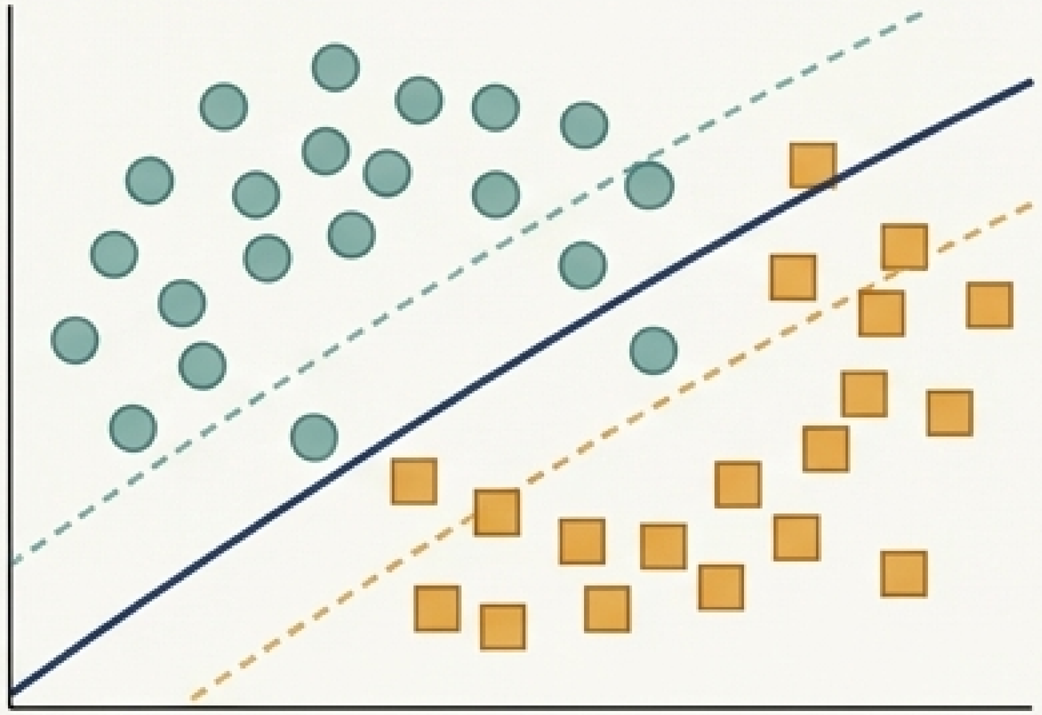
$$0 \leq \alpha_i \leq C$$

Large C



High penalty for errors. Aims to classify every point correctly, risking a narrow margin and overfitting (as in Exercise 3a).

Small C



Low penalty for errors. Prioritizes a wider, simpler margin, leading to better generalization (as in Exercise 3b).

An Unconstrained Perspective: The Hinge Loss

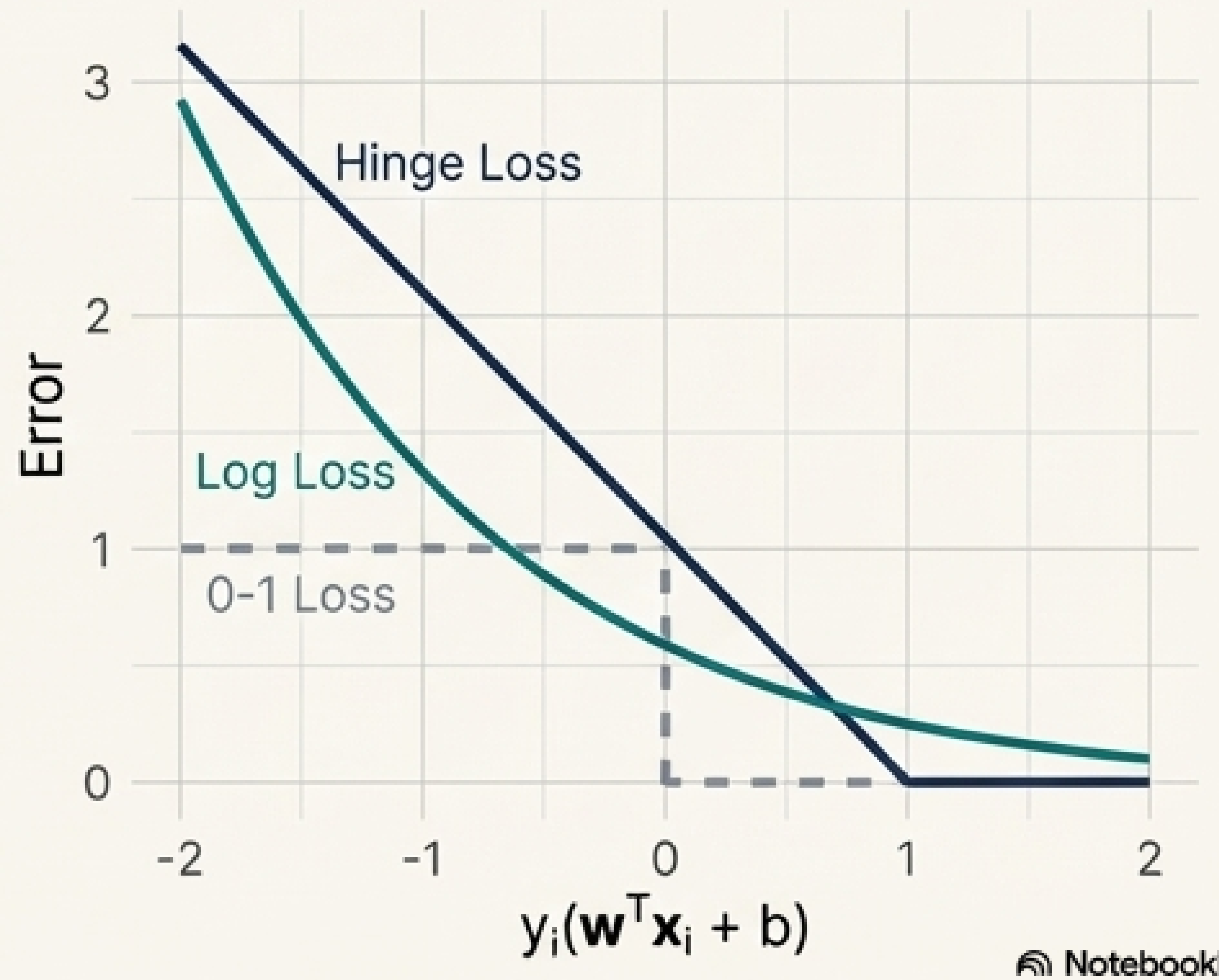
The soft-margin optimization problem is equivalent to the following unconstrained problem:

$$\min_{(\mathbf{w},\mathbf{b})} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \right\}$$

The term $\max(0, 1 - z)$ is the Hinge Loss. It is zero for points correctly classified outside the margin ($z \geq 1$) and applies a linear penalty for points that violate the margin ($z < 1$).

Connection to Logistic Regression (from Exercise 7):

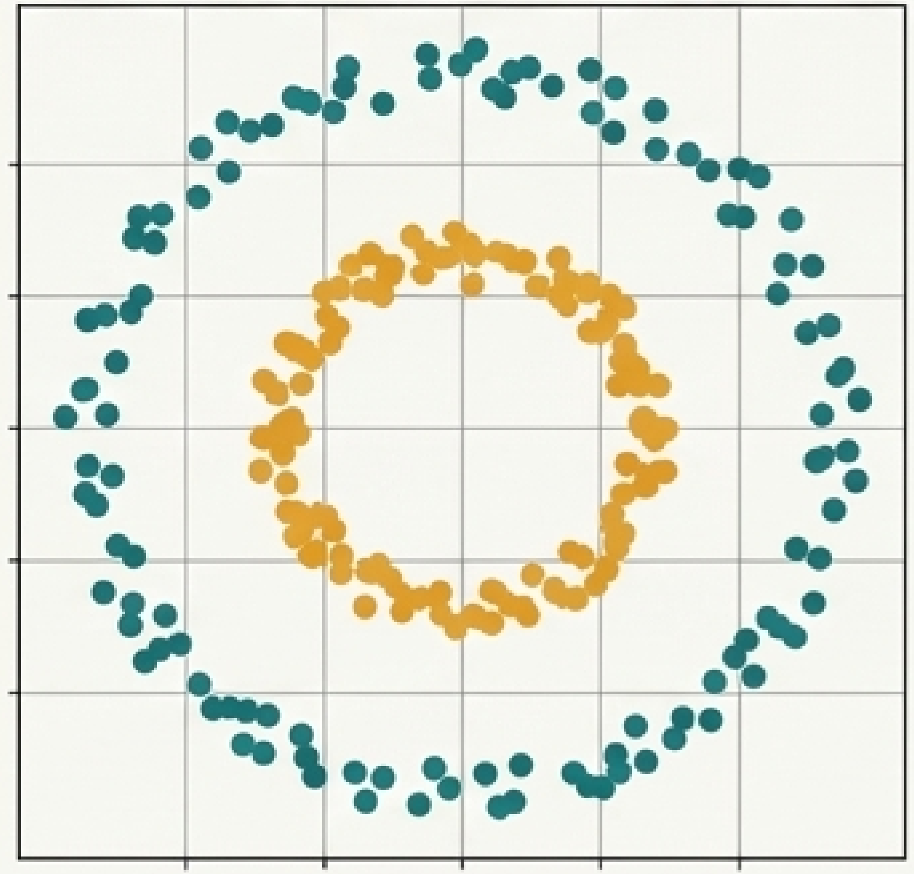
- Both SVM and L2-regularized Logistic Regression combine an L2 regularization term ($\lambda||\mathbf{w}||^2$) with a loss function.
- SVM uses the Hinge Loss, while Logistic Regression uses the Log Loss ($\ln(1 + \mathrm{e}^{-z})$).
- Both are convex upper bounds on the 0-1 classification error. However, Hinge Loss is “sparse” because it incurs zero penalty for well-classified points, whereas Log Loss always incurs a small penalty.



Beyond Linearity: The Kernel Trick

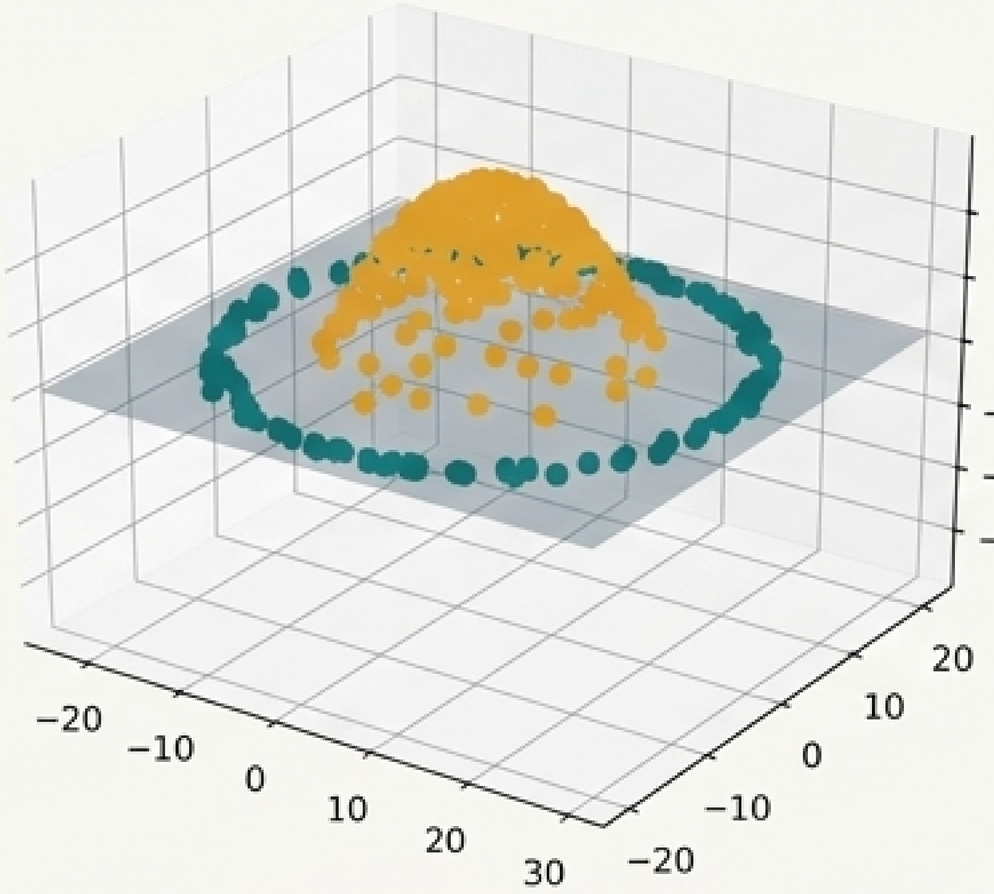
The Problem: Many real-world datasets are not separable by a linear boundary.
The Strategy: Map the data to a higher-dimensional feature space where it becomes linearly separable. Let this mapping be $\mathbf{x} \to \varphi(\mathbf{x})$.
The Challenge: Computing $\varphi(\mathbf{x})$ can be very expensive, and the feature space can even be infinite-dimensional.
The Key Insight: In the SVM dual formulation, the data $\mathbf{x}_i$ only appears in the form of inner products: $\mathbf{x}_i^T \mathbf{x}_j$.
The Kernel Trick: We can replace the inner product in the original space $\mathbf{x}_i^T \mathbf{x}_j$ with a kernel function $\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)$. This function efficiently computes the inner product in the high-dimensional feature space, $\varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$, without ever explicitly calculating the mapping $\varphi(\mathbf{x})$.

Original 2D Space: Non-Linear Separation



$\mathbf{x} \to \varphi(\mathbf{x})$

High-Dimensional Feature Space: Linear Separation



Choosing Your Lens: Examples of Kernels

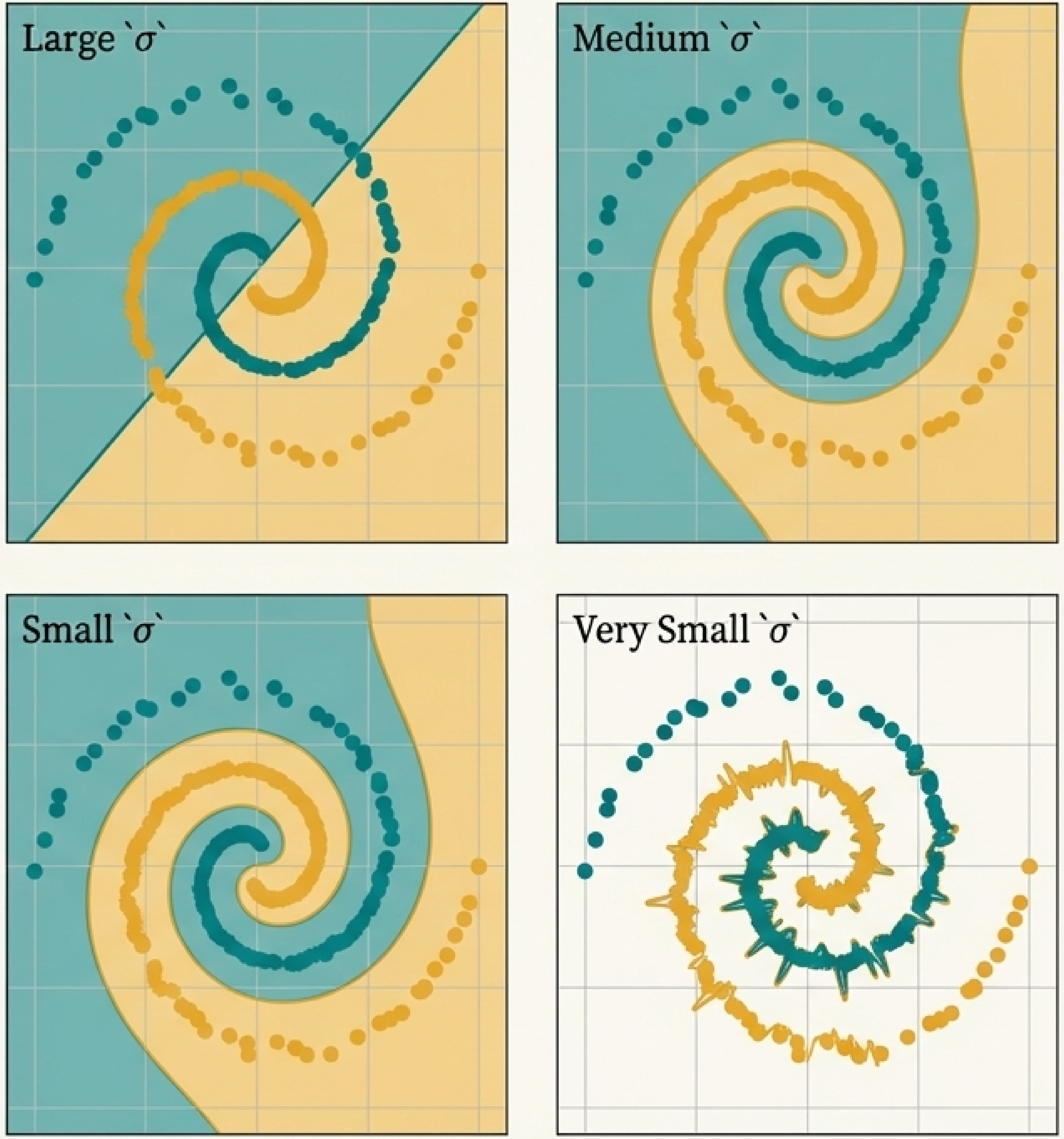
Polynomial Kernel: $k(a, b) = (a^T b + c)^d$. Creates polynomial decision boundaries of degree d.

Gaussian (RBF) Kernel: $k(a, b) = \exp(-||a - b||^2 / 2\sigma^2)$.

- This is a powerful, universal kernel corresponding to an *infinite-dimensional* feature space.
- The hyperparameter σ (sigma) controls the 'width' of the kernel's influence.

The effect of `$\sigma$`:

- **Small `$\sigma$`:** The decision boundary becomes highly localized and complex, wrapping around individual points, which risks overfitting. As $\sigma \to 0$, it can perfectly separate any finite set of points (**Exercise 4c**).
- **Large `$\sigma$`:** The influence of each point is broader, resulting in a smoother, more generalized, and near-linear decision boundary.



The Theory of Kernels: Validity and Construction

The Condition for Validity

A function $k(x, z)$ is a valid kernel if and only if it corresponds to an **inner product** in some feature space.

Mercer's Theorem

A function k is a valid kernel if the **Gram matrix** $\mathbf{K}$, where $K_{ij} = k(x_i, x_j)$, is **symmetric** and **positive semi-definite** for any set of points $\{x_1, \ldots, x_N\}$.

Constructing New Kernels

We can build valid kernels from simpler ones using a set of closure properties (as used in proofs in **Exercises 4b** and **11**):

1. **Sum:** $k_1 + k_2$ is a kernel.
2. **Scaling:** $c \cdot k_1$ is a kernel (for $c > 0$).
3. **Product:** $k_1 * k_2$ is a kernel.
4. **Mapping:** $k_3(\phi(x_1), \phi(x_2))$ is a kernel.

Putting It All Together: Prediction and Practice

1. Classifying a New Point x

The decision function depends only on the support vectors (indexed by the set S) and the chosen kernel:

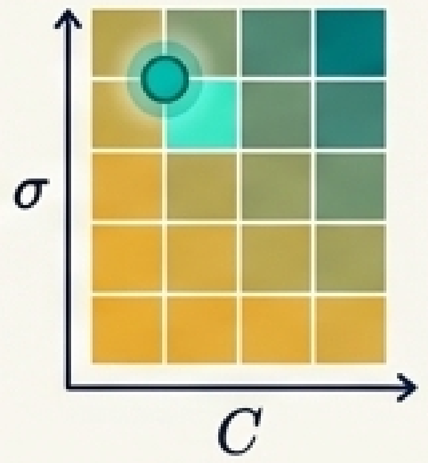
$$\boxed{h(x) = \text{sign}\left(\sum_{j \in S} \alpha_j y_j k(x_j, x) + b\right)}$$

2. Hyperparameter Tuning

SVM performance is highly sensitive to the choice of:

- The penalty parameter C.
- The kernel function (e.g., Linear, Polynomial, RBF).
- Kernel-specific parameters (e.g., σ for Gaussian, d for polynomial).

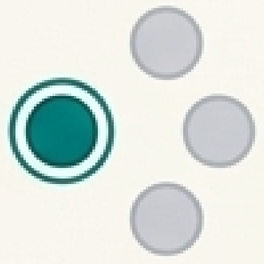
Best Practice: Use cross-validation to search for the optimal combination of these hyperparameters.



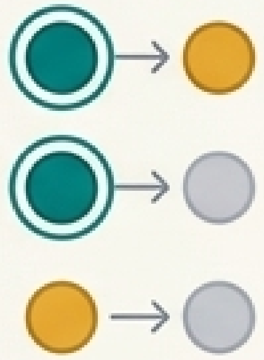
3. Handling Multiple Classes

Since SVM is a binary classifier, common strategies are:

- **One-vs-Rest:** Train K separate classifiers, each distinguishing one class from all the others.



- **One-vs-One:** Train $K(K-1)/2$ classifiers for every possible pair of classes and use majority voting.



The SVM Story: A Synthesis

Core Principle: Maximum Margin.

SVM is not just a separator; it's a maximum margin classifier, a principle motivated by strong theoretical guarantees for generalization.

The Power of the Dual Formulation.

The dual perspective is essential. It introduces sparsity via **support vectors**, enables the computational magic of the **kernel trick**, and results in a convex optimization problem that guarantees a unique global optimum (as proven in **Exercise 8**).

Flexibility for a Messy World.

- **Soft Margin (via `C`):** Provides robustness to noise and overlap by creating a trade-off between margin width and classification error.
- **Kernels (via `k(x,z)`):** Break the linear barrier by implicitly projecting data into high-dimensional spaces, allowing for powerful and highly non-linear decision boundaries.

In Essence.

SVM is a powerful, versatile, and theoretically sound algorithm for modern classification tasks.

Thank You