Week 11: Support Vector Machines

Maximum Margin Classification

2025

Learning Objectives

By the end of this week, you will be able to:

1. **Understand** the maximum margin principle and its geometric interpretation
2. **Derive** the soft margin SVM optimization problem
3. **Apply** the kernel trick to create non-linear decision boundaries
4. **Compare** different kernel functions (linear, polynomial, RBF, sigmoid)
5. **Tune** hyperparameters C and gamma for optimal performance
6. **Identify** support vectors and understand their role
7. **Extend** SVM to regression (SVR) and multi-class problems

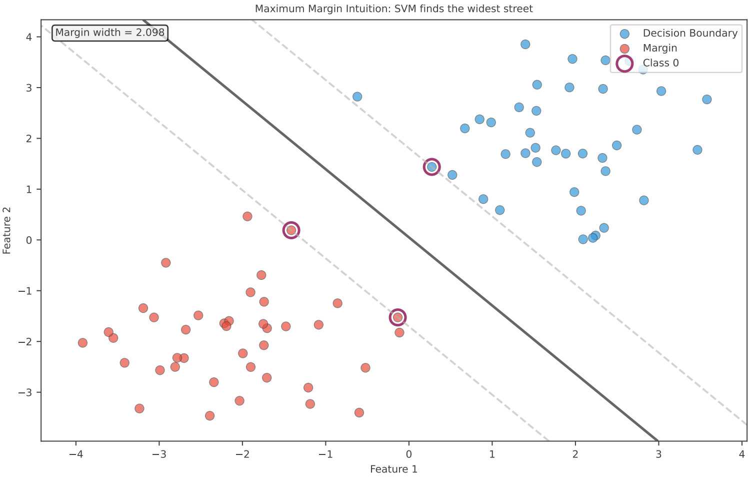
---

Support Vector Machines: Elegant theory meets practical power

Part I: Maximum Margin Principle

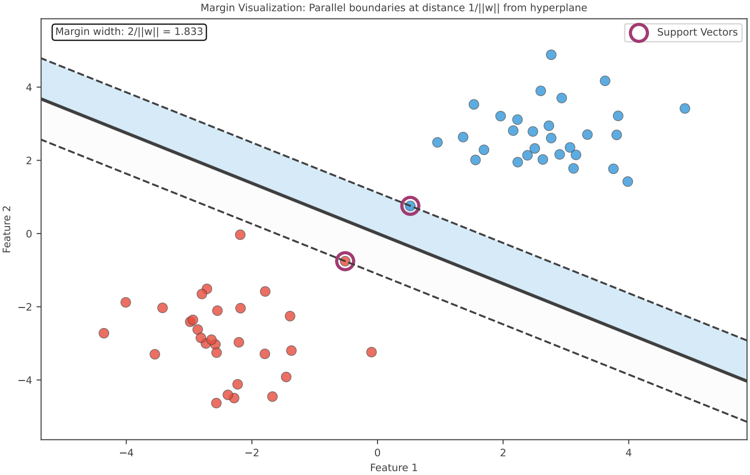
Maximize the gap between classes

Maximum Margin Intuition



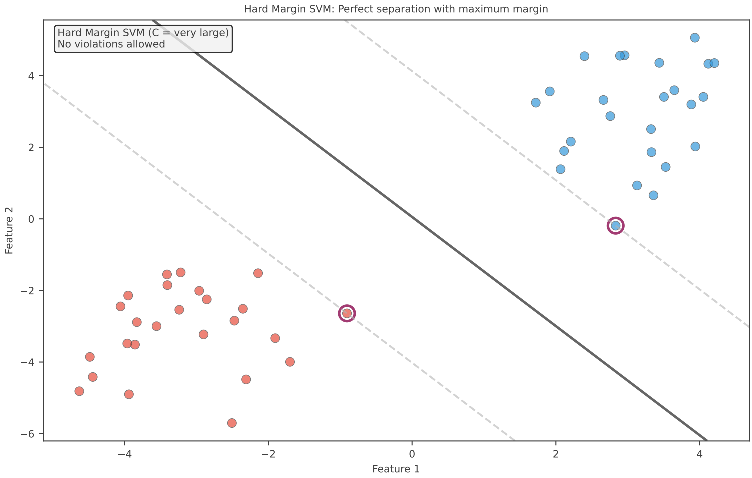
Among all separating hyperplanes, choose the one with maximum margin

Margin Visualization



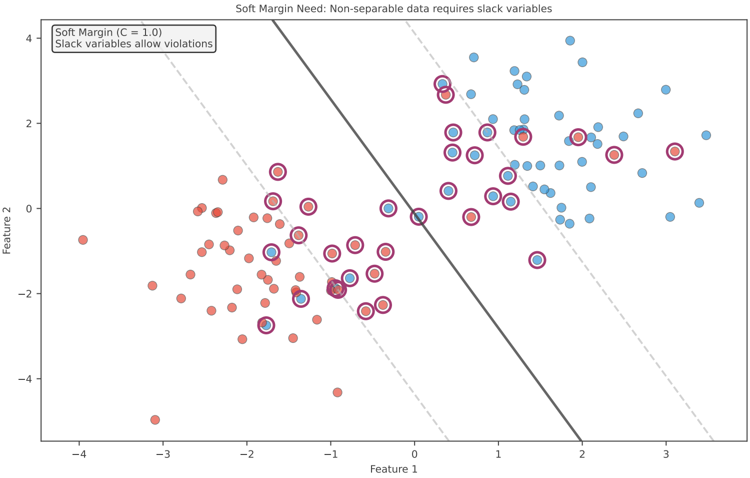
Margin: Perpendicular distance from hyperplane to nearest points

Hard Margin SVM



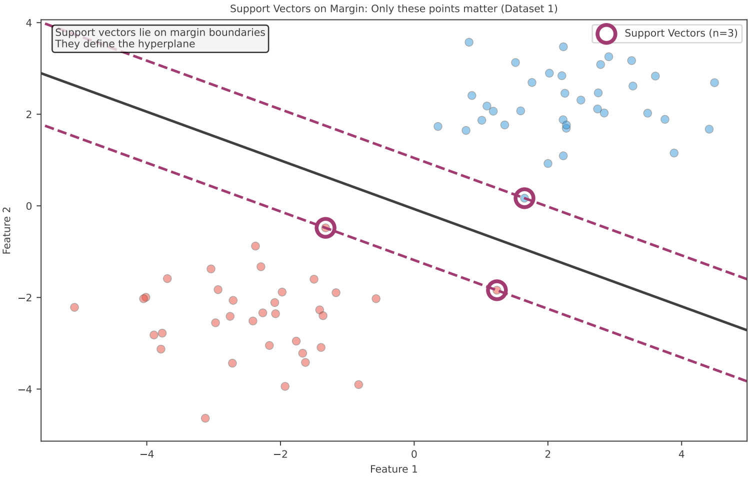
Hard margin: Perfect separation required, all points outside margin

Why We Need Soft Margin



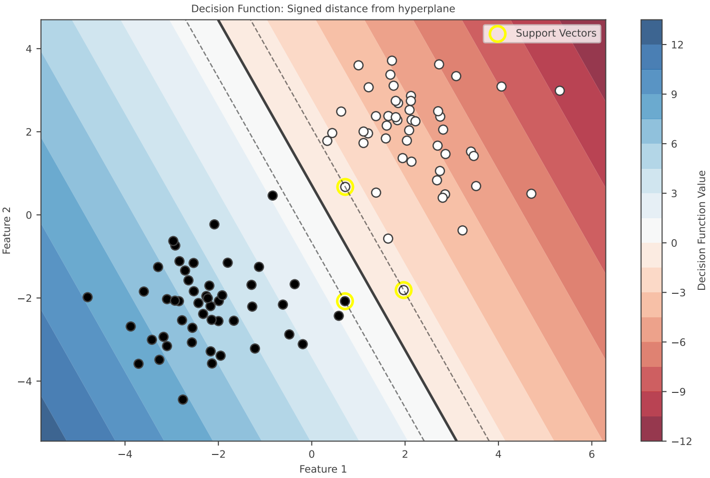
Real data rarely perfectly separable: Allow some violations with penalty

Support Vectors on Margin



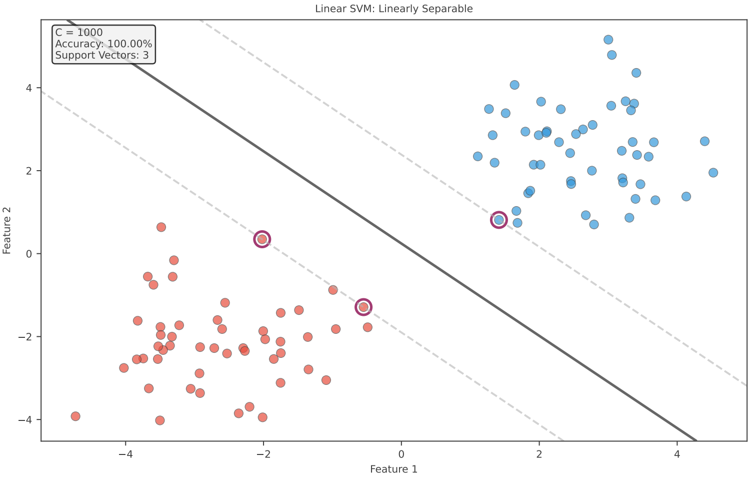
Support vectors: Points on or inside margin that define the hyperplane

Decision Function Values



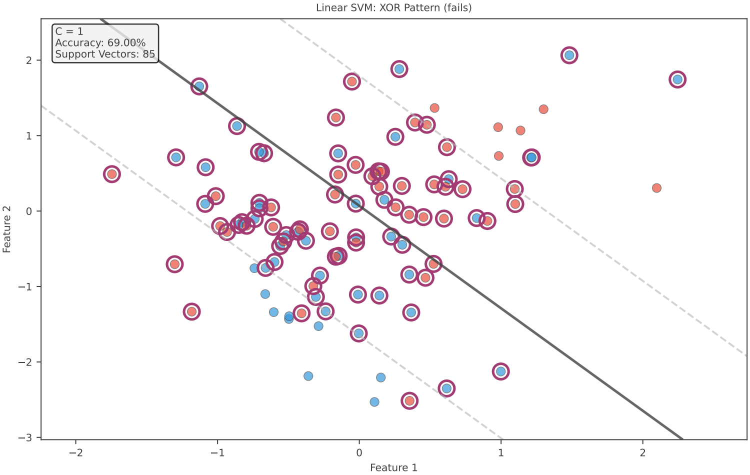
Signed distance from hyperplane indicates confidence

Linear SVM: When It Works



Linear SVM excels on linearly separable or near-separable data

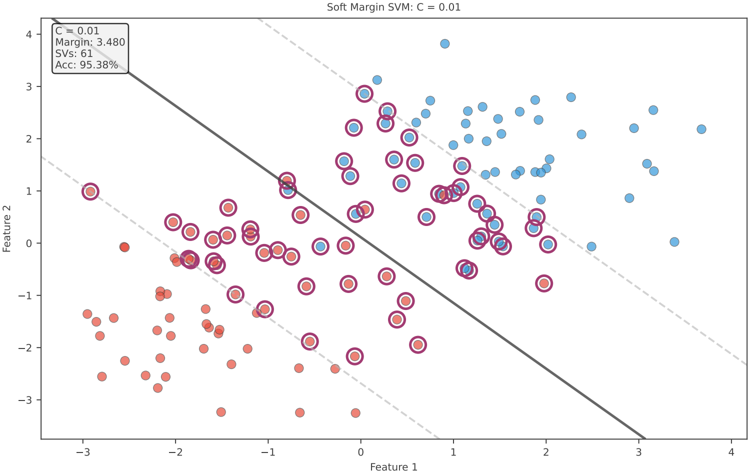
Linear SVM: When It Fails



Linear boundary cannot solve XOR problem: Need non-linear solution

C Parameter Effect

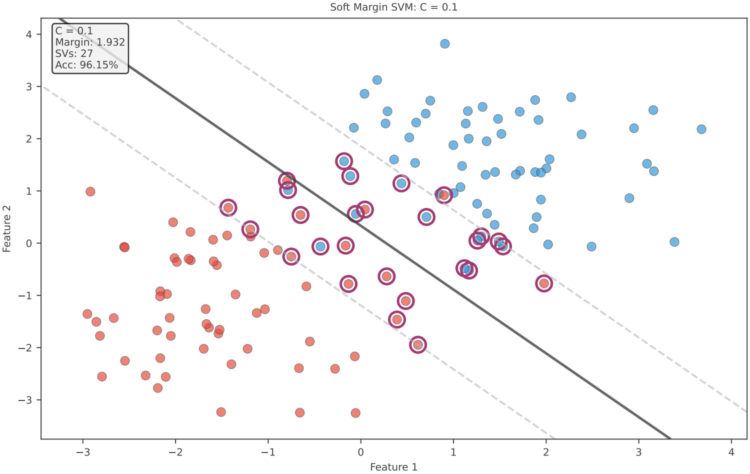
C = 0.01 (Very Soft Margin)



Very soft margin: Many support vectors, high tolerance for violations

C Parameter Effect

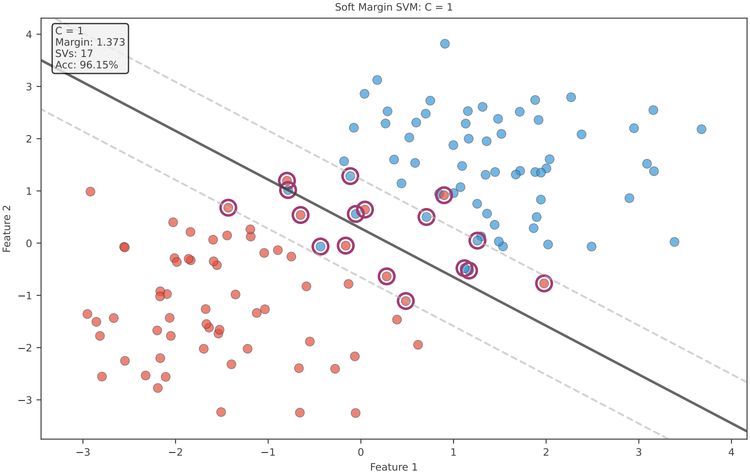
C = 0.1 (Soft Margin)



Soft margin: Flexible boundary, tolerates some misclassifications

C Parameter Effect

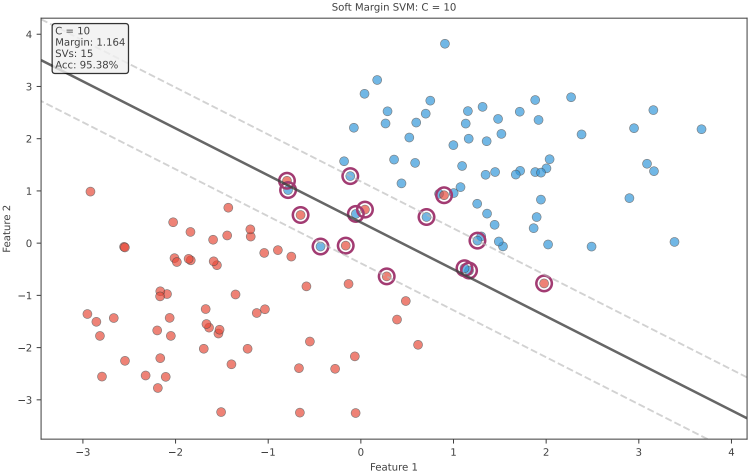
C = 1 (Balanced Margin)



Balanced margin: Default setting, moderate tolerance

C Parameter Effect

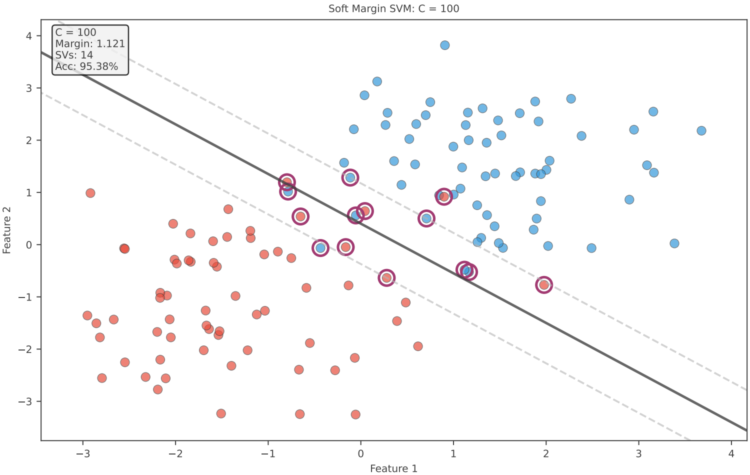
C = 10 (Stricter Margin)



Stricter margin: Fewer violations allowed, more support vectors on margin

C Parameter Effect

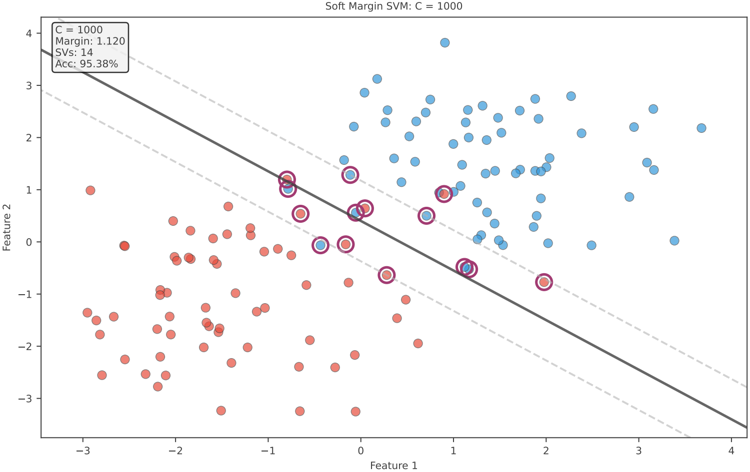
C = 100 (Hard Margin)



Hard margin: Very strict, approaches perfect separation

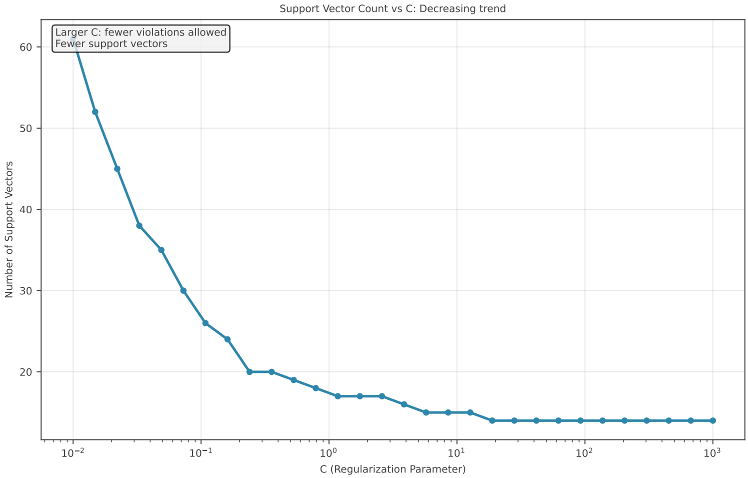
C Parameter Effect

C = 1000 (Very Hard Margin)



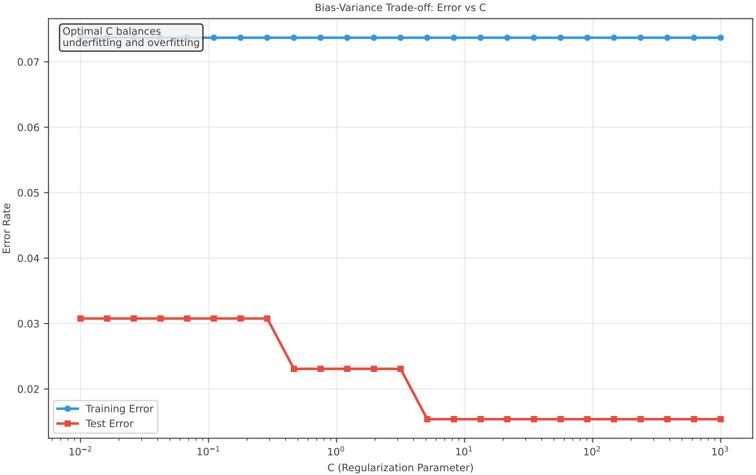
Very hard margin: Minimal violations, risk of overfitting

Support Vectors vs C



Fewer support vectors as C increases: Fewer violations allowed

Bias-Variance Trade-Off



C controls bias-variance: Small C = high bias, Large C = high variance

Why We Need Kernels

Problem

- Linear boundaries cannot solve non-linear problems
- XOR, circles, spirals are not linearly separable
- Need curved, complex decision boundaries

Approach

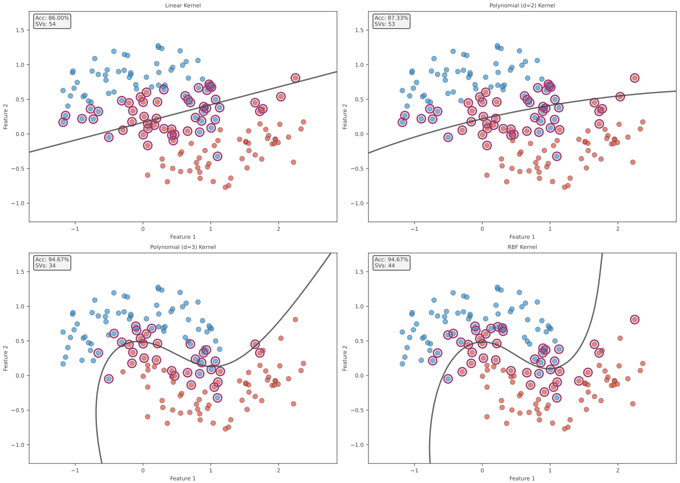
- Map data to higher-dimensional space
- Linear separation may exist in transformed space

Solution

- Kernel functions enable non-linear boundaries
- Implicit feature mapping via kernel trick

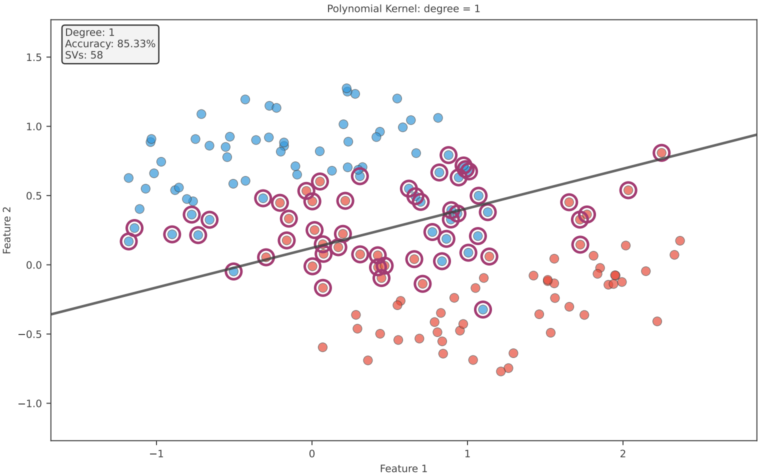
Kernels solve the non-linearity problem without explicit high-dimensional computation

Kernel Comparison



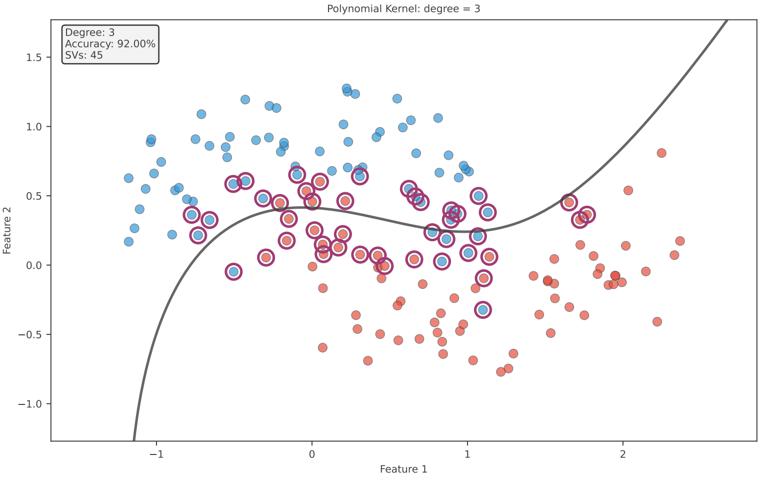
Different kernels create dramatically different decision boundaries

Polynomial Kernel: Degree 1 (Linear Boundary)



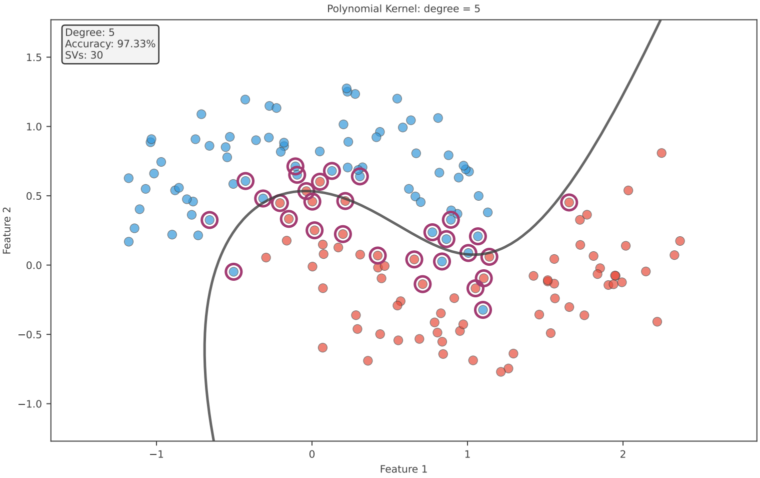
Degree 1 is equivalent to linear kernel: straight decision boundary

Polynomial Kernel: Degree 3 (Curved Boundary)



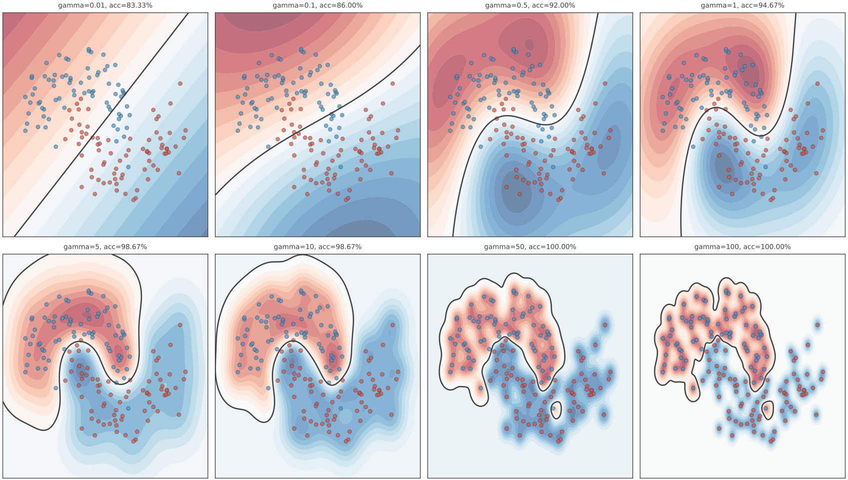
Degree 3 creates moderately complex curved boundaries

Polynomial Kernel: Degree 5 (Complex Boundary)



Higher degrees create very complex boundaries: flexible but risk overfitting

RBF Kernel: Gamma Comparison



Gamma controls RBF complexity: Small = smooth, Large = complex

The Kernel Trick: Feature Space Mapping

Problem

- Some patterns cannot be separated by any linear boundary
- Example: Circular patterns in 2D

Approach

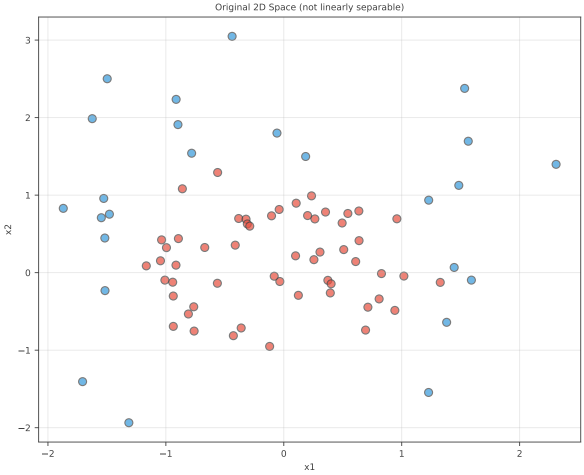
- Map to higher dimensions where linear separation exists
- Example: $(x_1, x_2) \to (x_1, x_2, x_1^2, x_1 x_2, x_2^2)$
- 2D circles become linearly separable in 5D

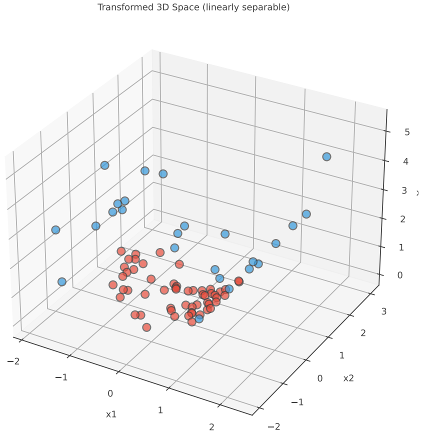
Solution: Kernel Trick

- Kernel function: $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$
- Compute inner product directly
- Never explicitly compute $\phi(\mathbf{x})$
- Efficient: $O(d)$ instead of $O(D)$ where $D \gg d$

Kernel trick: Implicit high-dimensional mapping without computational cost

Feature Space Mapping





Kernel trick: Implicitly map to higher dimensions without computing explicitly

Part II: Mathematical Foundations

Formalize the maximum margin concept

SVM Optimization Problem

Hard Margin (Primal)
Minimize:

$$\frac{1}{2}\|\mathbf{w}\|^2$$

Subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i$$

Margin Width

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

Maximizing margin = Minimizing $\|\mathbf{w}\|^2$

Soft Margin (Practical)
Minimize:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

Subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

C controls trade-off:

- Large C: Hard margin
- Small C: Soft margin

Soft margin allows misclassifications with penalty

Hard vs Soft Margin

Hard Margin vs Soft Margin SVM

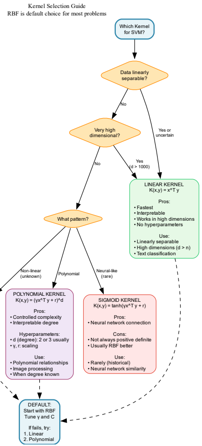
HARD MARGIN SVM	
Requirement	Data must be linearly separable No points in margin allowed
Objective	Maximize margin min 1/2 w squared
Constraints	y_i times w^T x_i + b greater or equal 1 for ALL points i
Support Vectors	Points exactly on margin y_i times w^T x_i + b = 1
C Parameter	Not applicable (infinite C implicitly)
Advantages	Unique solution Maximum separation Simple formulation
Disadvantages	FAILS if not separable Sensitive to outliers Not robust Rarely applicable
Use When	Perfect separation Clean data No noise

SOFT MARGIN SVM	
Requirement	Works with any data Allows margin violations
Objective	Maximize margin + penalize violations min 1/2 w squared + C sum of slack
Constraints	y_i times w^T x_i + b greater or equal 1 - slack_i slack_i greater or equal 0
Support Vectors	Points on margin OR inside alpha_i greater than 0
C Parameter	Controls trade-off Large C to hard margin Small C to soft margin
Advantages	Always has solution Robust to outliers Handles noise Practical
Disadvantages	Requires tuning C More hyperparameters Less unique depends on C
Use When	Real-world data always Noise present Overlapping classes Default choice

In practice: ALWAYS use soft margin (set finite C)
Hard margin is theoretical concept, soft margin is practical necessity

Soft margin is practical necessity for real-world data

Kernel Selection Guide - Visualization



Decision tree for choosing appropriate kernel based on data characteristics

Kernel Functions

Linear Kernel

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$$

Polynomial Kernel

$$K(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}^T \mathbf{y} + r)^d$$

RBF (Gaussian) Kernel

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$$

Sigmoid Kernel

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\gamma \mathbf{x}^T \mathbf{y} + r)$$

Kernel Trick
Decision function:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

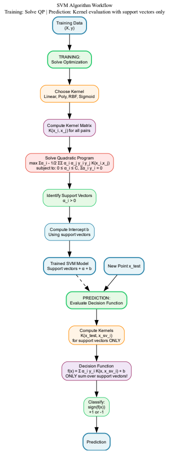
Only sum over **support vectors**!

Properties

- Linear: Fast, interpretable
- Polynomial: Controlled complexity
- RBF: Most flexible, default choice
- Sigmoid: Neural network-like

Kernels enable non-linear boundaries without explicit feature mapping

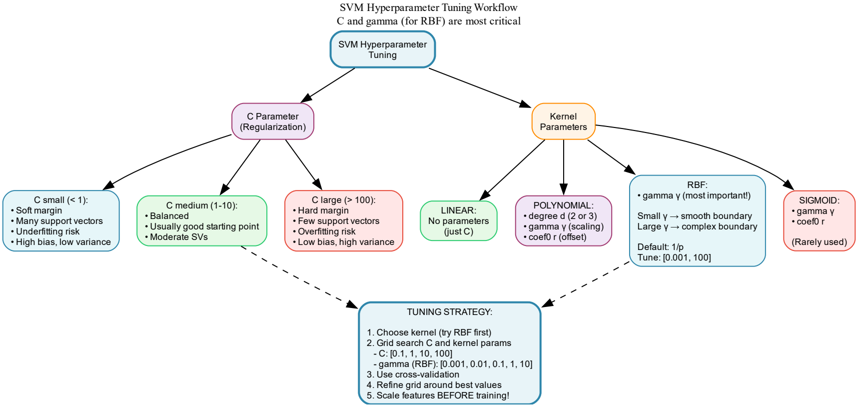
SVM Algorithm



Training: Solve QP — Prediction: Kernel evaluation with support vectors only

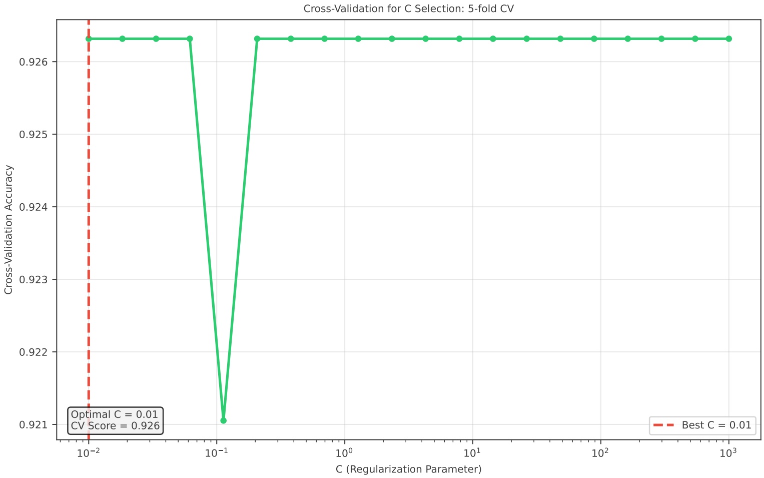
Part III: Hyperparameters & Practical Use

Tuning and application

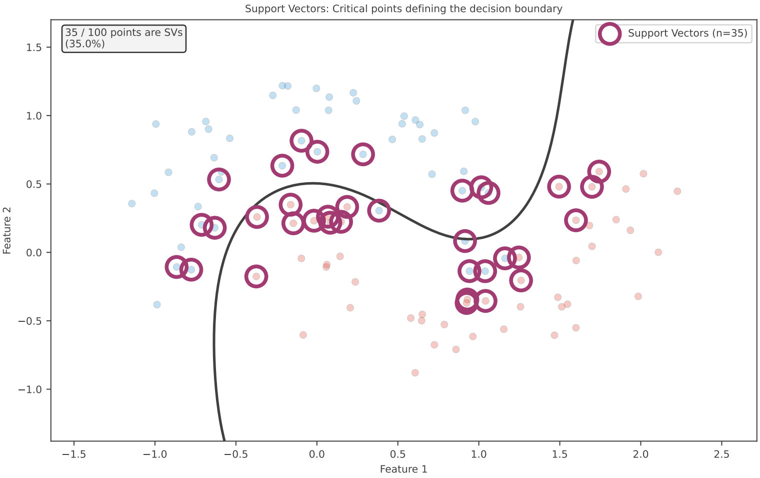


C and gamma (for RBF) are most critical hyperparameters

Cross-Validation for C

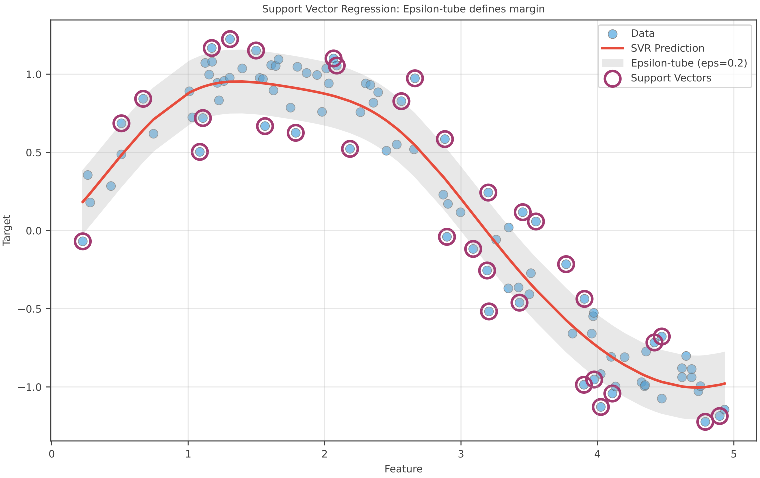


Use CV to find optimal C value



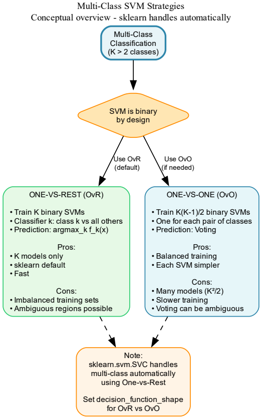
Support vectors are the critical points that define the margin

Support Vector Regression (SVR)



SVR: Epsilon-insensitive loss creates margin for regression

Multi-Class SVM



SVM naturally binary, extended to multi-class via One-vs-Rest or One-vs-One

Summary: Key Concepts

Core Ideas

- Maximum margin principle
- Support vectors define boundary
- Soft margin with C parameter
- Kernel trick for non-linearity

Optimization

- Quadratic programming
- Convex optimization
- Global optimum guaranteed

Kernels

- Linear: Default, interpretable
- Polynomial: Controlled complexity
- RBF: Most flexible, default choice
- Sigmoid: Neural network-like

Hyperparameters

- C: Margin softness
- gamma: RBF complexity
- degree: Polynomial complexity

SVM: Powerful, elegant, kernel-based classification and regression

Looking Ahead: Week 12

Next Week: Naive Bayes
Shift from discriminative to generative models:

- Probabilistic classification
- Bayes theorem application
- Gaussian, Multinomial, Bernoulli variants
- Independence assumption
- Text classification
- Fast training and prediction

Key Difference
SVM models boundaries. Naive Bayes models class distributions.

From maximum margin to maximum likelihood

Thank you

Questions?