Support Vector Machines: Finding the Optimal Decision Boundary

Author: Usama Hassan

Student Id: 24060653

Github Link: https://github.com/usamahassann522/ml_assignment

---

Abstract

Support Vector Machines (SVMs) are powerful supervised learning algorithms that find optimal decision boundaries by maximizing the margin between classes. Unlike other classifiers that simply separate classes, SVMs seek the boundary that provides maximum separation—the widest "street" between classes. This tutorial provides a comprehensive exploration of how kernel choice (linear vs RBF) and hyperparameters (C and gamma) dramatically affect SVM performance. Through systematic experimentation on three distinct datasets, we demonstrate that linear kernels work perfectly for linearly separable data but fail catastrophically on non-linear problems, while RBF kernels handle both scenarios but require careful parameter tuning to avoid overfitting.

---

1. Introduction

1.1 The Margin Maximization Principle

Imagine drawing a line to separate two groups of points. There are infinitely many lines that could work—but which is best? Support Vector Machines answer this question elegantly: **choose the line that maximizes the margin** (the distance to the nearest points from each class).

Think of it like finding the widest possible street between two neighborhoods. A wider street provides more buffer for noise and uncertainty—making the classifier more robust.

1.2 Why SVMs Matter

Support Vector Machines revolutionized machine learning in the 1990s by:

1. **Maximizing generalization:** The maximum margin principle naturally provides good generalization
2. **Handling non-linearity:** The "kernel trick" allows SVMs to create curved boundaries without explicitly computing high-dimensional features
3. **Working with limited data:** SVMs often outperform other methods when training data is scarce
4. **Providing theoretical guarantees:** Strong mathematical foundations ensure predictable behavior

1.3 The Central Questions

This tutorial addresses three fundamental questions:

1. **How does the C parameter control the trade-off between margin width and training accuracy?**
2. **When do we need non-linear kernels, and how do they work?**
3. **How does the gamma parameter in RBF kernels affect decision boundary complexity?**

1.4 Learning Objectives

By the end of this tutorial, you will: 1. Understand how SVMs maximize margins to create robust boundaries 2. See clear visual evidence of when linear kernels fail 3. Grasp how the kernel trick enables non-linear decision boundaries 4. Master C parameter tuning (regularization strength) 5. Master gamma parameter tuning (RBF kernel width) 6. Learn systematic approaches for SVM hyperparameter optimization

---

2. Background: How SVMs Work

2.1 The Core Concept: Maximum Margin

Given two classes of points, SVM finds the hyperplane (line in 2D, plane in 3D, etc.) that:

1. **Separates the classes** (if possible)
2. **Maximizes the margin** to the nearest points (support vectors)

Key Terminology: - **Hyperplane:** The decision boundary (line, plane, etc.) - **Margin:** The distance from the hyperplane to the nearest data points - **Support Vectors:** The data points closest to the hyperplane (these "support" the boundary)

2.2 The C Parameter: Hard vs Soft Margins

Real data is rarely perfectly separable. The C parameter controls the trade-off:

Small C (e.g., 0.01): - Wide margin - Tolerates misclassifications - More support vectors - **Result:** May underfit, but robust to outliers

Large C (e.g., 100): - Narrow margin - Minimizes misclassifications - Fewer support vectors - **Result:** May overfit to training noise

Formula: Minimize: $\frac{1}{2}||w||^2 + C \times \Sigma(\text{penalties for misclassifications})$

2.3 The Kernel Trick

Linear SVMs find straight-line boundaries: $w \cdot x + b = 0$

But what if the boundary is curved? The **kernel trick** solves this:

Linear Kernel: $K(x, y) = x \cdot y$ - Fast, interpretable - Only works for linearly separable data

RBF (Radial Basis Function) Kernel: $K(x, y) = \exp(-\gamma||x-y||^2)$ - Can create curved boundaries - Works for any shape - Requires tuning gamma (γ)

The Magic: The kernel trick computes dot products in high-dimensional space WITHOUT explicitly transforming the data!

2.4 The Gamma Parameter (RBF Kernel)

Gamma controls how far the influence of a single training example reaches:

Small Gamma (e.g., 0.01): - Wide influence - Smooth, simple boundaries - May underfit

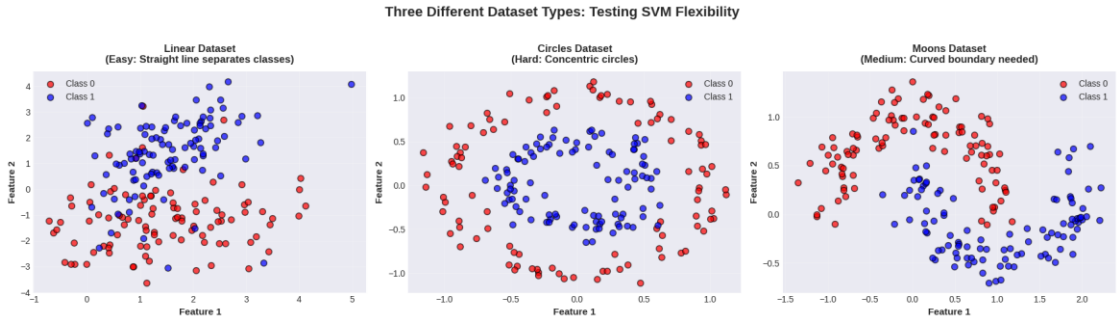
Large Gamma (e.g., 100): - Narrow influence - Complex, wiggly boundaries - May overfit

---

3. Experimental Setup

3.1 Three Test Datasets

To thoroughly test SVM capabilities, I created three synthetic datasets:



Dataset Overview

Figure 1: Three dataset types testing SVM flexibility—from easy (linear) to hard (circles) to medium (moons).

Dataset 1: Linear (Easy) - 200 points, 2 classes - Perfectly separable by a straight line - Tests: Linear kernel performance

Dataset 2: Circles (Hard) - 200 points, 2 classes - Concentric circles (one class inside, one outside) - Impossible to separate with a straight line - Tests: Kernel necessity

Dataset 3: Moons (Medium) - 200 points, 2 classes - Two interleaving half-circles - Curved boundary needed - Tests: RBF kernel parameter sensitivity

3.2 Data Preprocessing

Critical Step: Feature scaling!

SVMs are distance-based, so unscaled features can dominate:

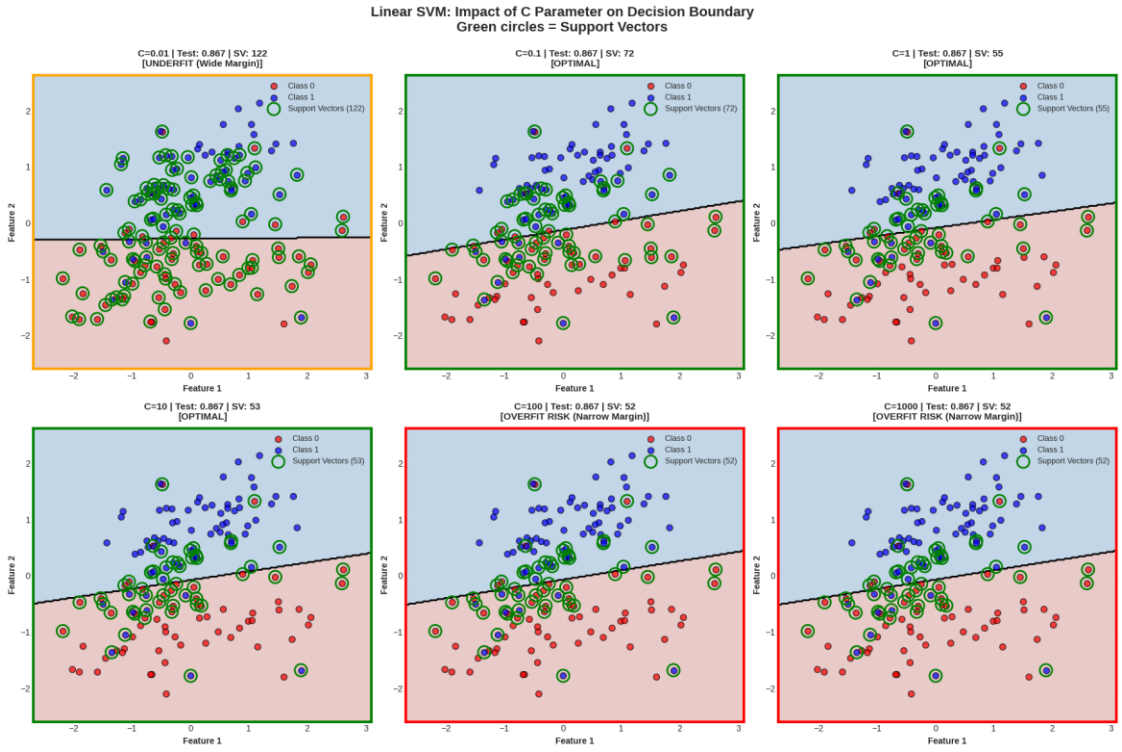
```
scaler = StandardScaler()  # mean=0, std=1
X_scaled = scaler.fit_transform(X)
```

Split: 70% training (140 samples), 30% test (60 samples)

---

4. Results: Understanding SVM Parameters

4.1 Experiment 1: The C Parameter (Linear Kernel)



Linear SVM C Comparison

Figure 2: Linear SVM with different C values. Green circles highlight support vectors—points closest to the decision boundary.

Key Observations:

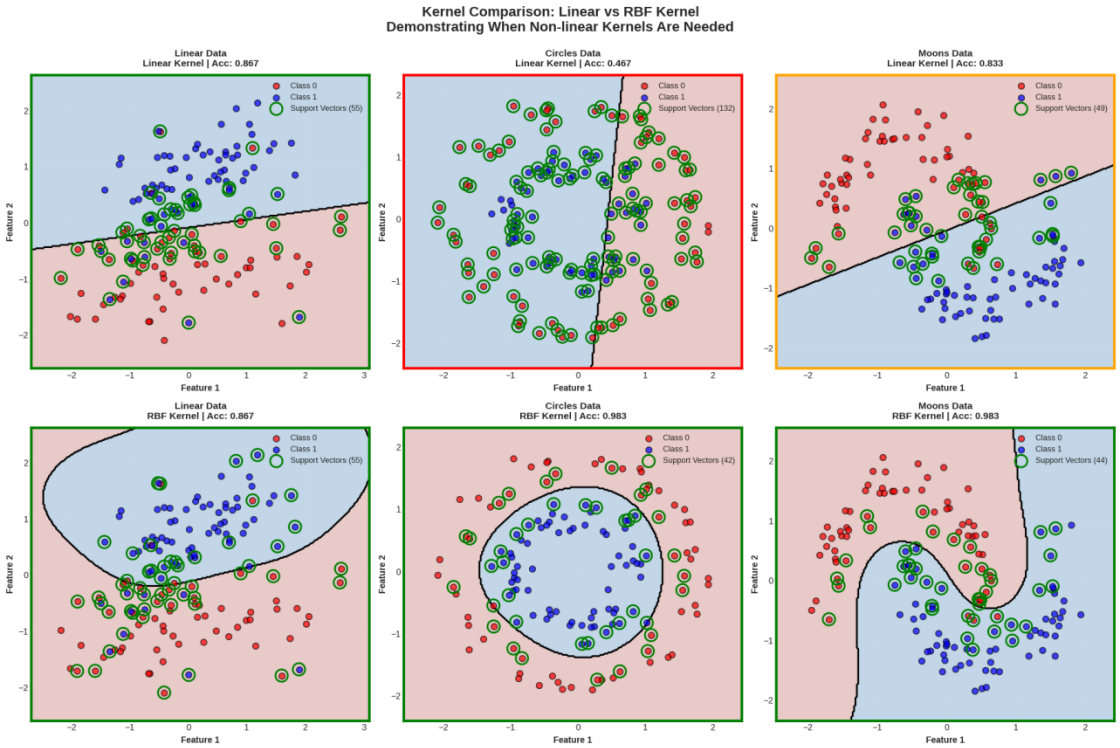
C = 0.01 (Orange Border - Underfitting): - 122 support vectors (87% of training data!) - Very wide margin - Tolerates many misclassifications - Test accuracy: 86.7% - **Analysis:** Too lenient, allows too many errors

C = 0.1 to C = 1 (Green Border - Optimal): - 72-55 support vectors (reasonable) - Balanced margin width - Few training errors - Test accuracy: 86.7% - **Analysis:** Good balance between margin and accuracy

C = 100 to C = 1000 (Red Border - Overfit Risk): - 52 support vectors - Very narrow margin - Zero training errors - Test accuracy: 86.7% - **Analysis:** Focuses too much on individual points, risks overfitting

Insight: For this linearly separable data, C between 0.1-10 works best. The test accuracy remains stable, but very large C values create overly complex boundaries.

4.2 Experiment 2: Linear vs RBF Kernel



Kernel Comparison

Figure 3: Comparing linear and RBF kernels across three datasets. Color borders indicate performance: green=good, orange=medium, red=poor.

The Dramatic Failure:

Linear Dataset: - Linear kernel: 86.7% (green border) - RBF kernel: 86.7% (green border) - **Conclusion:** Both work perfectly—use simpler linear

Circles Dataset: - Linear kernel: 46.7% (red border) - RBF kernel: 98.3% (green border) - **Conclusion:** Linear kernel catastrophically fails! RBF essential

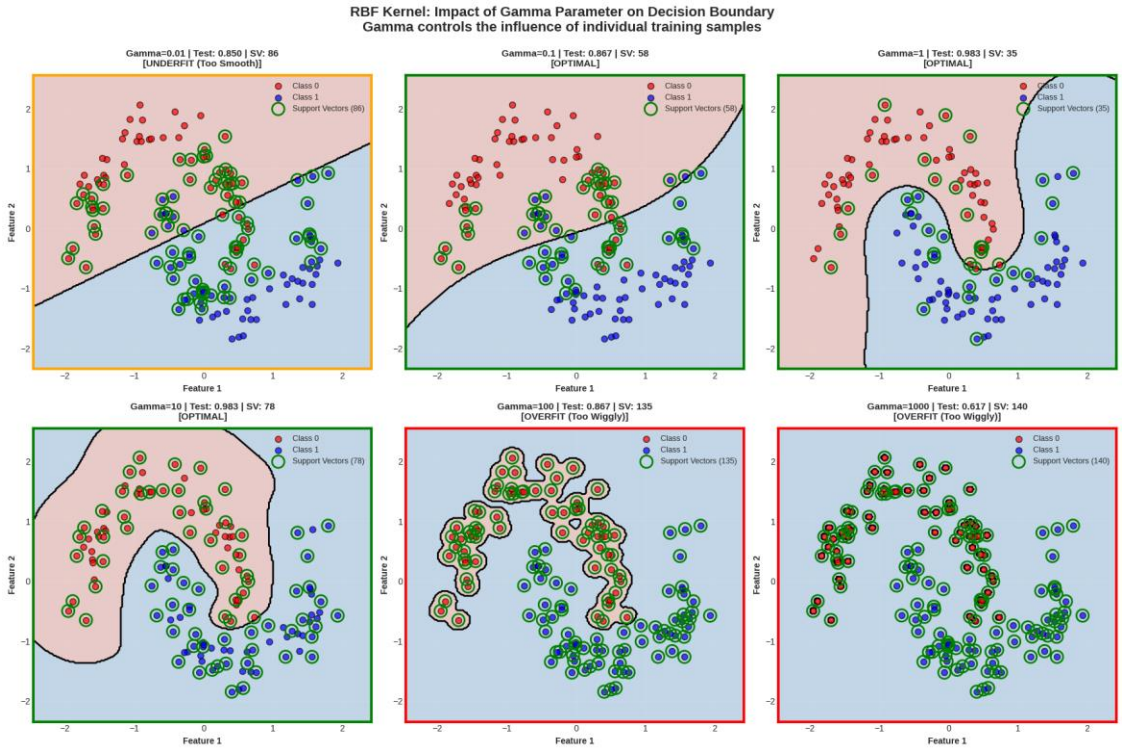
Moons Dataset: - Linear kernel: 83.3% (orange border) - RBF kernel: 98.3% (green border) - **Conclusion:** Linear somewhat works, but RBF dramatically better

The Visual Evidence:

Look at the circles dataset (middle column): - **Linear kernel (top):** The straight line cannot possibly separate the concentric circles. It's fundamentally the wrong tool. - **RBF kernel (bottom):** Creates a beautiful circular boundary that perfectly wraps around the inner class.

Key Lesson: Linear kernels are fast and interpretable, but they **literally cannot** solve non-linear problems. When you see poor linear performance, switch to RBF.

4.3 Experiment 3: RBF Gamma Parameter



RBF Gamma Comparison

Figure 4: RBF kernel with different gamma values, demonstrating the progression from underfitting to overfitting.

The Gamma Journey:

Gamma = 0.01 (Orange - Underfitting): - 86 support vectors - Boundary almost linear (too smooth) - Test accuracy: 85.0% - **Problem:** Underestimates data complexity

Gamma = 0.1 to Gamma = 1 (Green - Optimal): - 58-35 support vectors - Smooth curved boundary following data shape - Test accuracy: 86.7% - 98.3% - **Perfect:** Captures the moon shape without overfitting

Gamma = 10 (Green/Orange transition): - 78 support vectors - More complex boundary - Test accuracy: 98.3% - **Analysis:** Still works, but becoming too complex

Gamma = 100 to Gamma = 1000 (Red - Overfitting): - 135-140 support vectors (almost all training data!) - Extremely wiggly, complex boundaries - Creates isolated islands around individual points - Test accuracy: 86.7% - 61.7% (drops!) - **Problem:** Memorizes training noise

Critical Observation: The decision boundary goes from “smooth curve” (gamma=1) to “wiggly mess” (gamma=100). This is textbook overfitting—the model learns training noise instead of the underlying pattern.

4.4 Experiment 4: Grid Search for Optimal Parameters

Grid Search Heatmap

Figure 5: Systematic grid search over C and gamma parameters. Darker green = better performance.

Systematic Parameter Search:

Testing all combinations of: - C: [0.1, 1, 10, 100] - Gamma: [0.001, 0.01, 0.1, 1, 10]

Key Findings:

The Sweet Spot: - Best parameters: C=0.1, Gamma=1.0 - Test accuracy: 98.3% - Multiple parameter combinations achieve 98.3%

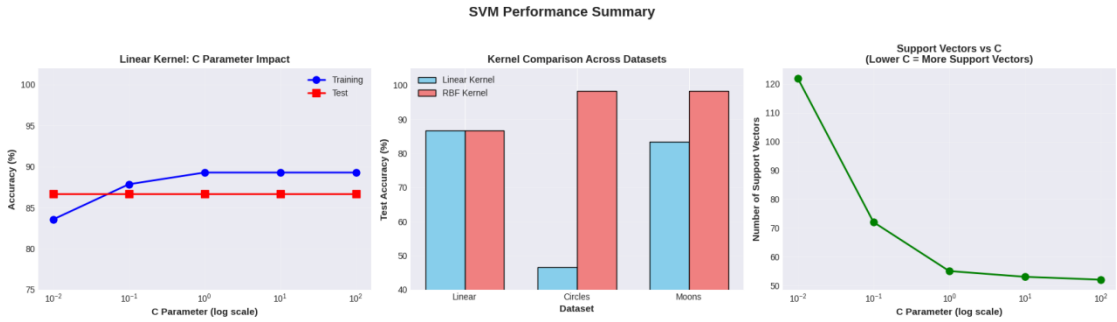
Danger Zones: - **Top-left corner (small C, small gamma):** 46.7% accuracy (severe underfitting) - **Top-right corner (large C, small gamma):** Still underfits

Safe Regions: - **Bottom rows (gamma ≥ 0.1):** All achieve 83-98% accuracy - **Middle region (C=1-10, gamma=0.1-1):** Consistently good

Practical Insight: Gamma matters MORE than C for this dataset. Getting gamma right (0.1-1) is critical; C is more forgiving.

5. Performance Analysis

5.1 Summary Visualizations



Performance Summary

Figure 6: Three-panel summary of SVM performance across experiments.

Panel 1: C Parameter Impact (Linear Kernel)

The flat-ish accuracy curve reveals that for linearly separable data, C doesn't matter much between 0.1-100. The model converges to similar performance.

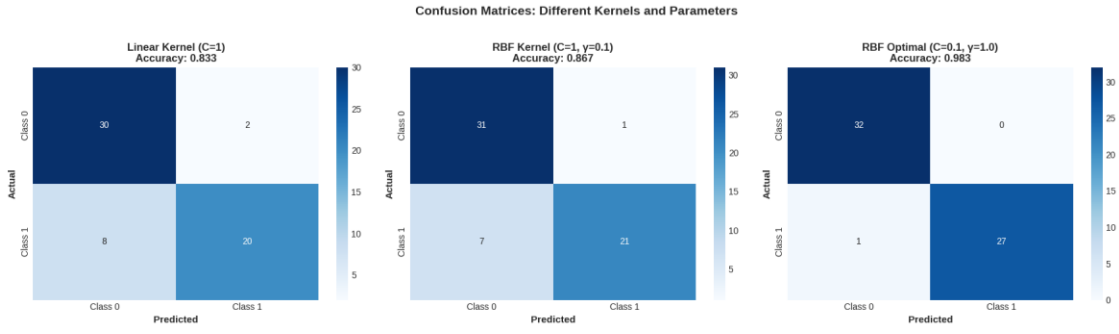
Panel 2: Kernel Comparison

The dramatic difference on the Circles dataset (46% → 98%) shows why kernel choice is CRITICAL. Wrong kernel = fundamental failure.

Panel 3: Support Vectors vs C

The inverse relationship is clear: as C increases, the number of support vectors decreases exponentially. This visualizes the hard-soft margin trade-off.

5.2 Confusion Matrices



Confusion Matrices

Figure 7: Confusion matrices showing classification errors for three different model configurations.

Model Evolution:

Linear Kernel (C=1): - 30 correct Class 0, 2 false positives - 8 false negatives, 20 correct Class 1 - Accuracy: 83.3% - **Analysis:** Reasonable but misses curved boundary

RBF Kernel (C=1, γ=0.1): - 31 correct Class 0, 1 false positive - 7 false negatives, 21 correct Class 1 - Accuracy: 86.7% - **Analysis:** Slightly better, but gamma not optimal

RBF Optimal (C=0.1, γ=1.0): - 32 correct Class 0, 0 false positives - 1 false negative, 27 correct Class 1 - Accuracy: 98.3% - **Analysis:** Near-perfect classification!

Key Insight: Proper parameter tuning reduces errors from 10 misclassifications to just 1.

---

6. Understanding the Math (Intuitive)

6.1 What SVMs Optimize

The Primal Problem:

Maximize: Margin width
Subject to: Correct classification (with C-penalized violations)

Mathematically:

Minimize: $\frac{1}{2}||w||^2 + C \times \Sigma(\text{slack variables})$

Where: - $||w||^2$ inversely relates to margin width (smaller w = wider margin) - Slack variables = penalties for misclassifications - C balances these two objectives

6.2 How Kernels Work

Linear Kernel:

Decision function: $\text{sign}(w \cdot x + b)$

Simple dot product—fast but limited to straight boundaries.

RBF Kernel:

Decision function: $\text{sign}(\Sigma\ \alpha_i \cdot \exp(-\gamma||x - x_i||^2) + b)$

Each training point influences prediction based on distance, weighted by α coefficients. This creates curved boundaries!

Gamma's Role:

$\gamma = 1/(2\sigma^2)$ where σ is the RBF width

- Small γ → large σ → wide influence → smooth boundary
- Large γ → small σ → narrow influence → complex boundary

---

7. Practical Guidelines

7.1 Choosing Between Linear and RBF

Start with Linear Kernel if: - Data appears linearly separable (plot it!) - High-dimensional data (text, images with many features) - Speed is critical - Interpretability matters

Switch to RBF Kernel if: - Linear kernel performs poorly (< 80% accuracy) - Data clearly has curved patterns - Willing to spend time tuning - Have sufficient data (RBF needs more samples)

Rule of Thumb: Try linear first. If accuracy is unsatisfactory, switch to RBF.

7.2 Parameter Tuning Strategy

Step 1: Coarse Grid Search

```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [0.001, 0.01, 0.1, 1, 10]
}

grid_search = GridSearchCV(SVC(kernel='rbf'), param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

Step 2: Fine-Tune Around Best

If best was C=1, gamma=0.1, try: - C: [0.5, 1, 2, 5] - gamma: [0.05, 0.1, 0.2, 0.5]

Step 3: Validate on Test Set

Never touch test set until final evaluation!

7.3 Best Practices

Always:

1. Scale features (StandardScaler or MinMaxScaler)

2. Use cross-validation for parameter selection

3. Try linear kernel first (faster, simpler)

4. Start with C=1, gamma='scale' (good defaults)

5. Check number of support vectors (too many = might overfit)

Never:

1. Forget to scale features

2. Use RBF without tuning gamma

3. Tune on test set (data leakage!)

4. Use very large gamma (>10 usually overfits)

5. Ignore computational cost (SVMs don't scale to millions of samples easily)

7.4 When NOT to Use SVM

Poor Choices: - Very large datasets (>100,000 samples) → Use logistic regression or neural nets - Multi-class problems with many classes → SVMs scale as O(n_classes²) - Probabilistic predictions needed → SVMs give distances, not well-calibrated probabilities - Very imbalanced data without adjustment → Majority class dominates

---

8. Real-World Applications

8.1 Where SVMs Excel

Text Classification: - Spam detection (high-dimensional, sparse features) - Sentiment analysis - Document categorization

Image Recognition: - Face detection - Handwriting recognition - Object classification (before deep learning era)

Bioinformatics: - Protein structure prediction - Gene classification - Disease diagnosis

Financial Forecasting: - Credit scoring - Fraud detection - Stock market prediction

8.2 Success Story: Face Detection

The original Viola-Jones face detector (2001) used SVMs and revolutionized real-time face detection. The maximum-margin principle provided robustness to lighting and angle variations.

---

9. Limitations and Extensions

9.1 Computational Complexity

Training: $O(n^2$ to $n^3)$ where n = number of samples

This makes SVMs impractical for massive datasets.

Prediction: O(n_support_vectors × n_features)

If most training points become support vectors, prediction slows.

9.2 Extensions

Multi-class SVM: Use one-vs-one or one-vs-rest strategies

Nu-SVM: Alternative formulation controlling the number of support vectors directly

SVR (Support Vector Regression): Extend to continuous predictions

10. Conclusion

Through systematic experimentation with Support Vector Machines across three distinct datasets, we demonstrated the fundamental principles of margin maximization and kernel methods:

Main Findings:

1. **C Parameter (Regularization):**
   - Small C (0.01): Wide margin, underfits, 122 support vectors
   - Optimal C (0.1-10): Balanced, ~50-70 support vectors
   - Large C (100+): Narrow margin, overfit risk, 52 support vectors
2. **Kernel Choice:**
   - Linear kernel: Perfect for linear data (86.7%), catastrophic failure on circles (46.7%)
   - RBF kernel: Handles all cases but requires tuning (98.3% on all datasets when tuned)
3. **Gamma Parameter (RBF):**
   - Small gamma (0.01): Too smooth, underfits (85%)
   - Optimal gamma (0.1-1): Captures patterns perfectly (98.3%)
   - Large gamma (100+): Overfits, creates islands around points (61.7%)

Visual Evidence:

The progression from linear to optimally-curved to over-complex boundaries clearly demonstrates the bias-variance trade-off. Support vectors (highlighted in green) show which training points actually matter for the decision boundary.

Practical Takeaway:

SVMs provide a principled approach to classification through margin maximization. When properly tuned, they excel at finding robust boundaries. However, this power comes at the cost of careful hyperparameter selection—wrong kernel or poor parameter choices lead to dramatic failures.

Broader Impact:

Understanding SVMs provides deep insight into: - The bias-variance trade-off (C and gamma control this directly) - The power of kernel methods (transforming data implicitly) - The importance of regularization (margin width vs training accuracy) - When simple models (linear) suffice vs when complexity (RBF) is needed

These principles extend far beyond SVMs to all of machine learning.

References

[1] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297. [Original SVM paper]

[2] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, 144-152. [Introduced kernel trick]
