Support Vector Machine (SVM) – Complete Beginner-to-Advanced Revision Guide

---

1. Introduction

SVM intuition – maximum margin hyperplane

Illustration: two classes separated by the maximum-margin hyperplane; circled points are support vectors.

1. Introduction

What is a Support Vector Machine?

A **Support Vector Machine (SVM)** is a **supervised learning algorithm** used for **classification and regression** tasks.

The core idea of SVM is to **find the optimal decision boundary (hyperplane)** that best separates data points of different classes by **maximising the margin**.

Key Intuition

- Decision boundary with **maximum margin**
- Only a subset of points matter → **support vectors**
- Powerful for **high-dimensional spaces**

Real-World Applications

- Face recognition
- Text classification (spam detection)
- Bioinformatics (gene classification)
- Handwriting recognition
- Financial risk modelling

---

2. Mathematical Foundations

Linear SVM geometry

Illustration: hyperplane, margin, and distance to the closest points.

2. Mathematical Foundations

2.1 Linear Separability

Assume binary labels:

$$y_i \in \{-1, +1\}$$

We want a hyperplane:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

Such that:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) > 0$$

2.2 Margin

Functional Margin

$$\gamma_i = y_i(\mathbf{w}^T\mathbf{x}_i + b)$$

Geometric Margin

$$\hat{\gamma}_i = \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Goal: Maximise the minimum geometric margin.

2.3 Canonical Form

We scale $\mathbf{w}, b$ such that:

$$\min_i y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$$

Then the margin becomes:

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

---

3. Hard-Margin SVM (Linearly Separable Case)

Hard margin SVM

Illustration: perfectly separable data with no margin violations.

3. Hard-Margin SVM (Linearly Separable Case) (Linearly Separable Case)

Optimization Problem

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

Subject to:

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

---

4. Soft-Margin SVM (Non-Separable Data)

Soft margin SVM with slack variables

Illustration: margin violations controlled by slack variables.

4. Soft-Margin SVM (Non-Separable Data) (Non-Separable Data)

Slack Variables

Introduce $\xi_i \geq 0$:

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

Optimization Problem

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

Where: - C controls bias–variance tradeoff

---

5. Dual Formulation (Why It Matters)

Lagrangian

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

Dual Problem

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

Subject to:

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \quad \alpha_i \geq 0$$

---

6. Kernel Trick

Kernel trick intuition

Illustration: mapping non-linearly separable data into higher-dimensional space.

6. Kernel Trick

Motivation

When data is **not linearly separable**, map it into a higher-dimensional space.

Instead of computing $\phi(x)$ explicitly:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Common Kernels

Linear

$$K(x_i, x_j) = x_i^T x_j$$

Polynomial

$$K(x_i, x_j) = (x_i^T x_j + c)^d$$

RBF (Gaussian)

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

Sigmoid

$$K(x_i, x_j) = \tanh(\alpha x_i^T x_j + c)$$

---

7. Decision Function

Support vectors and decision boundary

Illustration: only support vectors influence the final decision boundary.

7. Decision Function

$$f(x) = \sum_{i \in SV} \alpha_i y_i K(x_i, x) + b$$

Only **support vectors** contribute.

---

8. Model Evaluation

Same metrics as logistic regression:

- Accuracy
- Precision
- Recall
- F1-score
- ROC-AUC

---

9. Interpretation

- Large $\|w\|$ → small margin
- Support vectors define the boundary
- Kernel choice defines feature space

---

10. Assumptions

- Data is separable (approximately)
- Kernel captures structure
- Features are scaled
- Independent samples

---

11. Common Pitfalls & Misconceptions

- ❌ Forgetting feature scaling
- ❌ Using RBF without tuning C and γ
- ❌ Assuming SVM gives probabilities (it doesn't by default)
- ❌ Too many support vectors → overfitting
- ❌ Interpreting coefficients in non-linear SVM

---

12. Python Implementation

12.1 From Scratch (Conceptual – Linear SVM)

```python
import numpy as np

# Hinge loss gradient descent (simplified)
def train_svm(X, y, lr=0.01, epochs=1000, C=1.0):
    m, n = X.shape
    w = np.zeros(n)
    b = 0

    for _ in range(epochs):
        for i in range(m):
            condition = y[i] * (np.dot(X[i], w) + b)
            if condition >= 1:
                w -= lr * w
            else:
                w -= lr * (w - C * y[i] * X[i])
                b += lr * C * y[i]

    return w, b
```

12.2 Using scikit-learn

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = SVC(kernel='rbf', C=1.0, gamma='scale')
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

12.3 Decision Boundary Visualization

SVM decision boundary examples

Illustration: linear vs RBF kernel decision boundaries.

12.3 Decision Boundary Visualization

```
import matplotlib.pyplot as plt
import numpy as np

plt.scatter(X[:,0], X[:,1], c=y)
plt.show()
```

---

13. Hyperparameter Tuning

- **C**: regularization strength
- **gamma**: RBF kernel width
- **kernel**: linear / poly / rbf

Use GridSearchCV.

---

14. When NOT to Use SVM

- Very large datasets (slow)

- Noisy data with overlapping classes
- When interpretability is critical

---

15. Best Practices

- Always scale features
- Start with linear kernel
- Tune C before gamma
- Use cross-validation

---

16. Summary

SVM is: - Margin-based - Kernel-powered - Robust in high dimensions - A cornerstone ML algorithm

Understanding SVM builds intuition for: - Convex optimization - Kernel methods - Advanced classifiers

---

End of Revision Guide