POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Support Vector Machines

Numerical Analysis for Machine Learning

Author: **Miguel Planas Díaz**

Student ID: 11071870
Mail: miguel.planas@mail.polimi.it
Advisor: Edie Miglio
Academic Year: 2024-25

Contents

1 | Introduction to Support Vector Machines (SVM)

Support Vector Machines (SVMs), introduced by Boser, Guyon, and Vapnik in 1992 [1], emerged as a powerful classification method rooted in statistical learning theory, primarily developed by Vladimir Vapnik and Alexey Chervonenkis. This framework gained prominence in the late 1990s, positioning SVMs as a leading classifier across various fields, including text analysis [2] and genomic data classification [3].

In 1997, Vapnik expanded SVM applications to include co-regression for predicting multiple related outputs, enhancing their capability to analyze variable relationships. The introduction of kernel functions further increased their flexibility, allowing SVMs to handle complex data types like graphs and sequences by projecting inputs into higher-dimensional spaces for improved class separation. In 1999, Schölkopf integrated principal component analysis (PCA) with SVMs, while Platt and Joachims developed Sequential Minimal Optimization (SMO) to optimize SVM performance efficiently.

One of the most compelling advantages of SVMs is their robustness in handling high-dimensional data, making them suitable for applications with large feature spaces. Unlike neural networks, they are not hindered by local optima, ensuring reliable performance across various datasets. SVMs also excel in managing the balance between model complexity and generalization through explicit control of regularization parameters. This feature allows practitioners to tailor the trade-off between margin width and classification error to specific problem requirements. Moreover, their ability to utilize custom kernel functions extends their reach to non-traditional data types like strings and trees, reinforcing their versatility.

However, the effectiveness of SVMs hinges on the careful selection of kernel functions and parameters. This tuning process, often reliant on cross-validation, can be computationally intensive and requires domain expertise. Despite these challenges, the elegant mathematical foundation and demonstrated success in diverse domains affirm the enduring relevance of SVMs in machine learning research and practice.

2 | Understanding the Functionality of SVM

Support Vector Machines (SVMs) are a powerful supervised learning technique primarily used for classification tasks. They operate by analyzing a dataset of observations, where each observation consists of two components:

- A **feature vector**:
$$x_i \in \mathbb{R}^n,\ i = 1, \ldots, N$$
- A corresponding **label**:
$$y_i \in \{-1, +1\}$$

The fundamental goal of SVMs is to determine a **hyperplane** that effectively separates observations into two categories: **positive samples** (+1) and **negative samples** (−1). As illustrated in Figure 2.1, this separation is achieved by maximizing the **margin** between the two classes, a central concept in SVM methodology.

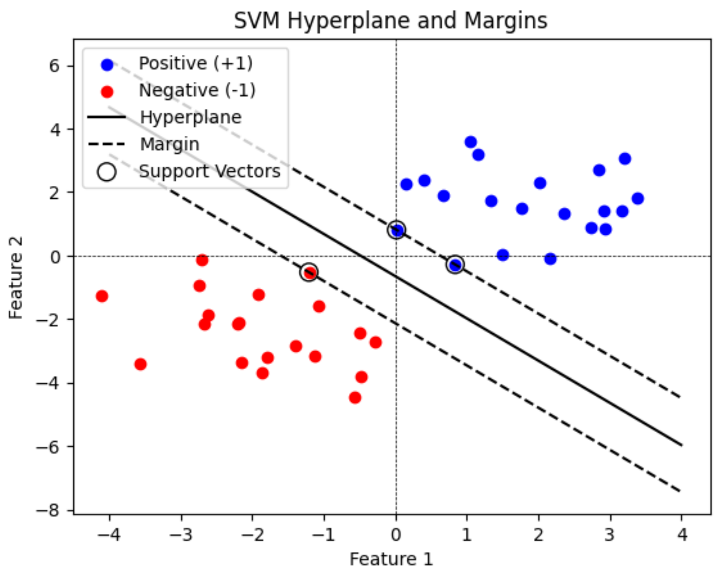

Figure 2.1: Illustration of the SVM hyperplane separating positive and negative samples. The margins are shown with dashed lines, and support vectors are highlighted.

2.1. Hyperplane and Margin in Support Vector Machines

A **hyperplane** in an n-dimensional space is a linear decision boundary that separates the space into two distinct half-spaces. It is mathematically defined as:

$$w^T x + b = 0,$$

- $w \in \mathbb{R}^n$ is the **normal vector** to the hyperplane, determining its orientation,
- $b \in \mathbb{R}$ is the **bias term**, shifting the hyperplane relative to the origin.

The position of any data point x_i relative to the hyperplane can be evaluated based on the sign of the function $w^T x_i + b$:

- **Points lying on the hyperplane** satisfy:
$$w^T x_i + b = 0.$$
- **Points on the positive side of the hyperplane** fulfill:
$$w^T x_i + b > 0.$$
- **Points on the negative side of the hyperplane** meet:
$$w^T x_i + b < 0.$$

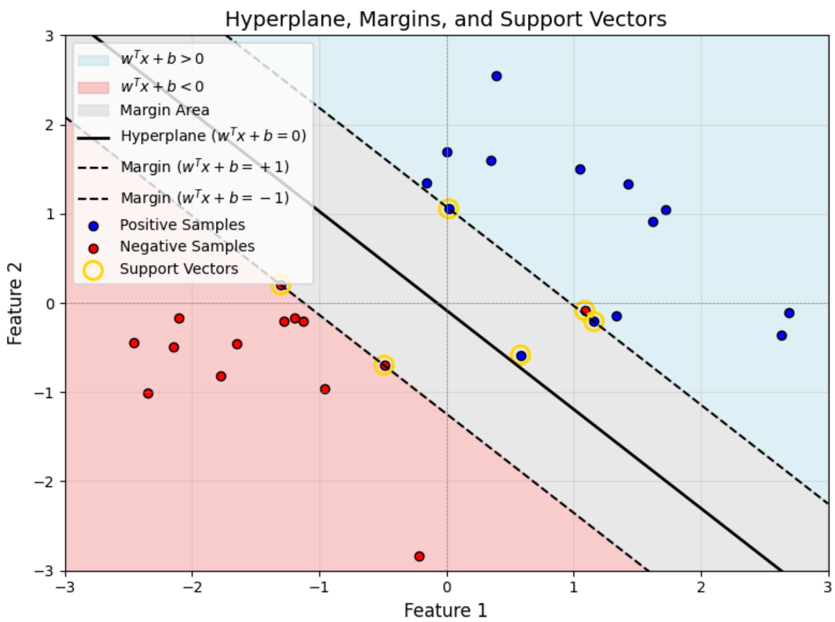

Figure 2.2

Figure 2.2 illustrates the concept of the hyperplane in a two-dimensional space. The regions where $w^T x + b > 0$ (positive side) and $w^T x + b < 0$ (negative side) are shaded, while the boundary itself is represented as the hyperplane $w^T x + b = 0$.

The Margin and Support Vectors

The **margin** is a critical concept in Support Vector Machines (SVM), defined as the distance between the hyperplane and the nearest data points from each class. These closest data points are referred to as the **support vectors**, as they play a fundamental role in determining the optimal positioning of the hyperplane.

The margin is mathematically given by:

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

where $\|w\|$ is the Euclidean norm of the normal vector w. By maximizing this margin, SVM aims to find the most **generalizable** decision boundary, ensuring better performance on unseen data.

Role of the Support Vectors The support vectors are the data points that lie closest to the hyperplane. They are critical because:

- The position of the hyperplane is entirely determined by the support vectors. Changing or removing any other data point has no effect on the decision boundary.
- These points define the margins, which are the boundaries where $w^T x + b = \pm 1$.
- They help SVM to achieve a balance between model complexity and generalization, as the margin maximization reduces overfitting.

Why Maximizing the Margin is Important Maximizing the margin is essential for the following reasons:

- A larger margin increases the confidence in predictions for new, unseen data points, as they are farther from the decision boundary.
- It reduces the risk of **overfitting** the model to the training data, making the classifier more robust.
- The SVM algorithm focuses only on the support vectors, ignoring points far from the decision boundary. This property makes SVM particularly efficient when dealing with high-dimensional data.

3 | Classification with SVM

Support Vector Machines (SVMs) are a versatile and powerful tool for classification tasks. This chapter explores SVM classification in the context of linearly and non-linearly separable problems, as well as non-linear SVM techniques for more complex scenarios.

3.1. Overview of SVM Classification

SVM classification problems can be broadly categorized into three types:

- **Linearly separable problems:** These problems assume the existence of a linear hyperplane that separates two classes without errors.
- **Non-linearly separable problems:** In these cases, some degree of misclassification is allowed to handle noise or overlapping data points.
- **Non-linear SVMs:** When the data cannot be separated even with a soft margin, more sophisticated techniques, such as kernel functions, are introduced to handle complex boundaries.

Real-world consideration: In practice, most datasets are not linearly separable, necessitating the use of non-linear SVMs or soft margin approaches to achieve satisfactory results.

3.2. Linearly Separable Problems

For linearly separable problems, the goal is to identify a **hyperplane** that maximizes the separation between two classes. This hyperplane is defined as:

$$w^T x + b = 0,$$

where w is the normal vector to the hyperplane and b is the bias term.

3.2.1. Classification Function

The classification function can be expressed as:

$$f(x_i) = \begin{cases} +1, & \text{if } w^T x_i + b > 0 \quad (y_i = +1), \\ -1, & \text{if } w^T x_i + b < 0 \quad (y_i = -1). \end{cases} \tag{3.1}$$

or equivalently:

$$f(x_i) = \text{sign}(w^T x_i + b). \tag{3.2}$$

Here, the function f classifies each data point into $+1$ or -1.

3.2.2. The Concept of Margin

To ensure robustness and generalization, the SVM algorithm identifies the hyperplane that maximizes the **margin**. The margin is defined as the distance between the hyperplane and the closest points from each class, which are known as the **support vectors**.

The margin d is defined as the distance between the two hyperplanes that separate the classes. For any training example, this relationship can be described as:

$$w^T x_i + b \leq -\frac{d}{2} \quad \text{if} \quad y_i = -1, \quad \text{and} \quad w^T x_i + b \geq +\frac{d}{2} \quad \text{if} \quad y_i = +1.$$

These conditions can be expressed in a single compact form:

$$y_i(w^T x_i + b) \geq \frac{d}{2}.$$

For support vectors, which lie exactly on the margin boundary, this inequality becomes an equality:

$$y_i(w^T x_i + b) = \frac{d}{2}.$$

If we scale w and b such that the margin $d/2$ equals 1, the equation simplifies to:

$$y_i(w^T x_i + b) = 1.$$

The perpendicular distance of a point x_i to the hyperplane can be computed as:

$$r = \frac{w^T x_i + b}{\|w\|}.$$

For support vectors, which are on the margin boundary, this distance becomes $\frac{1}{\|w\|}$. Thus, the total margin d is defined as:

$$d = 2r = \frac{2}{\|w\|}.$$

3.2.3. Optimization Problem

The SVM optimization problem can thus be defined as:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1 \, \forall i.$$

This is a quadratic programming problem (QP) that ensures the margin is maximized while correctly classifying all training points.

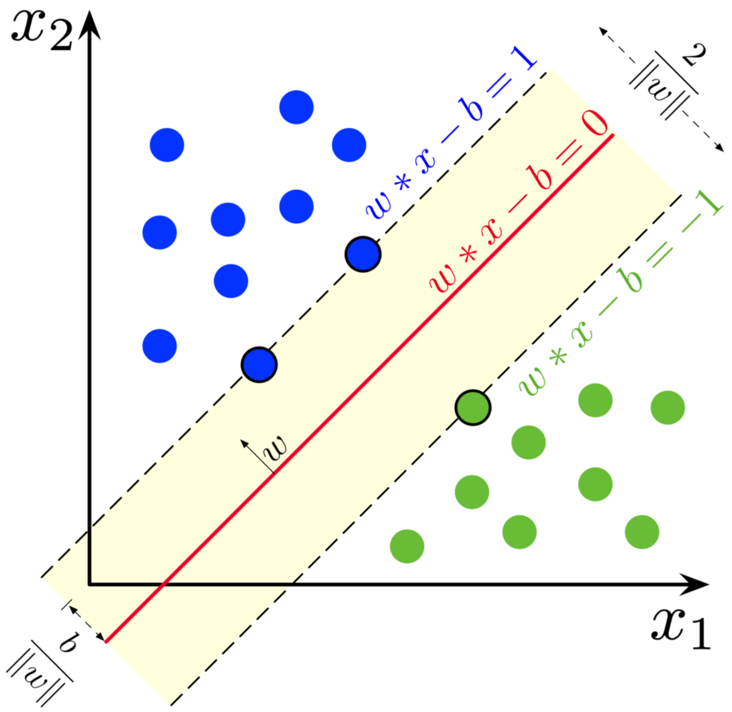

Figure 3.1: SVM margins. Source: [4].

Dual Problem and Lagrange Multipliers

To simplify the problem, we introduce the Lagrange multipliers α_i for each training sample and reformulate the primal problem into its dual form. The Lagrangian is defined as:

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{I} \alpha_i \left[y_i(w^T x_i + b) - 1 \right],$$

where $\alpha_i \geq 0$ for all i.

Taking the partial derivatives of L_p with respect to w and b, and setting them to zero:

$$\frac{\delta L_p}{\delta w} = 0 \quad \Longrightarrow \quad w = \sum_{i=1}^{I} \alpha_i y_i x_i,$$

$$\frac{\delta L_p}{\delta b} = 0 \quad \Longrightarrow \quad \sum_{i=1}^{I} \alpha_i y_i = 0.$$

Substituting w into the Lagrangian yields the dual problem:

$$L_D = \sum_{i=1}^{I} \alpha_i - \frac{1}{2} \sum_{i=1}^{I} \sum_{j=1}^{I} \alpha_i \alpha_j y_i y_j x_i^T x_j,$$

subject to:

$$\sum_{i=1}^{I} \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0 \, \forall i.$$

This dual form is a concave optimization problem that can be efficiently solved using quadratic programming techniques.

Optimal Hyperplane and Classification Function

The solution to the dual problem provides the Lagrange multipliers α_i. The optimal hyperplane can be expressed as:

$$w = \sum_{i=1}^{I} \alpha_i y_i x_i,$$

where only the training points with $\alpha_i > 0$ contribute to w. These points are called **support vectors**.

The bias term b can be computed as:

$$b = y_k - w^T x_k \quad \text{for any support vector}\, x_k \,\text{with}\, \alpha_k > 0.$$

Finally, the decision function for classifying a new point x is given by:

$$f(x) = \text{sign} \left(\sum_{i=1}^{I} \alpha_i y_i x_i^T x + b \right).$$

Key Insight

- Only support vectors ($\alpha_i > 0$) influence the hyperplane.
- Non-support vectors have $\alpha_i = 0$ and do not affect the decision boundary.

This formulation highlights the efficiency of SVM, as the model depends only on a subset of the training data, making it robust and scalable.

3.3. Non-Linearly Separable Problems

In real-world applications, data is often not perfectly linearly separable. This means that no single hyperplane can separate the two classes without error. To address this, Support Vector Machines (SVMs) introduce the concept of **soft margins**, which allow for some degree of misclassification.

The function to be optimized in the soft margin Support Vector Machine (SVM) is given by:

$$\text{Maximize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Here:

- w is the normal vector to the hyperplane,
- b is the bias term,
- C is the regularization parameter that controls the trade-off between maximizing the margin and minimizing classification errors,
- ξ_i are the slack variables that allow for margin violations.

This formulation ensures that the SVM can handle non-linearly separable data by introducing slack variables ξ_i to permit some misclassifications. The goal is to maximize the margin while keeping violations as small as possible, controlled by the parameter C.

3.3.1. Soft Margin Concept

The idea of soft margins is to relax the strict separation constraints by introducing **slack variables** ξ_i, which measure how much each data point violates the margin. The modified constraints become:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, N,$$

where y_i represents the class label ($+1$ or -1), x_i is the feature vector, and w and b define the hyperplane.

The objective function to minimize now includes a penalty term for misclassified points:

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

where:

- $\|w\|$ is the Euclidean norm of the normal vector w,
- C is a regularization parameter that controls the trade-off between maximizing the margin and allowing for misclassification,
- ξ_i are the slack variables that measure the degree of margin violation.

3.3.2. Interpretation of the Parameter C

The regularization parameter C plays a crucial role in the soft margin SVM:

- A **large** C places a higher penalty on misclassification, resulting in a smaller margin and fewer errors, but it may lead to overfitting.
- A **small** C allows for a larger margin and tolerates more misclassifications, improving generalization but at the cost of training accuracy.

As shown in Figure 3.2, a high value of C tightly fits the data, resulting in a narrow margin, while a lower C increases the margin width by allowing a few misclassifications.

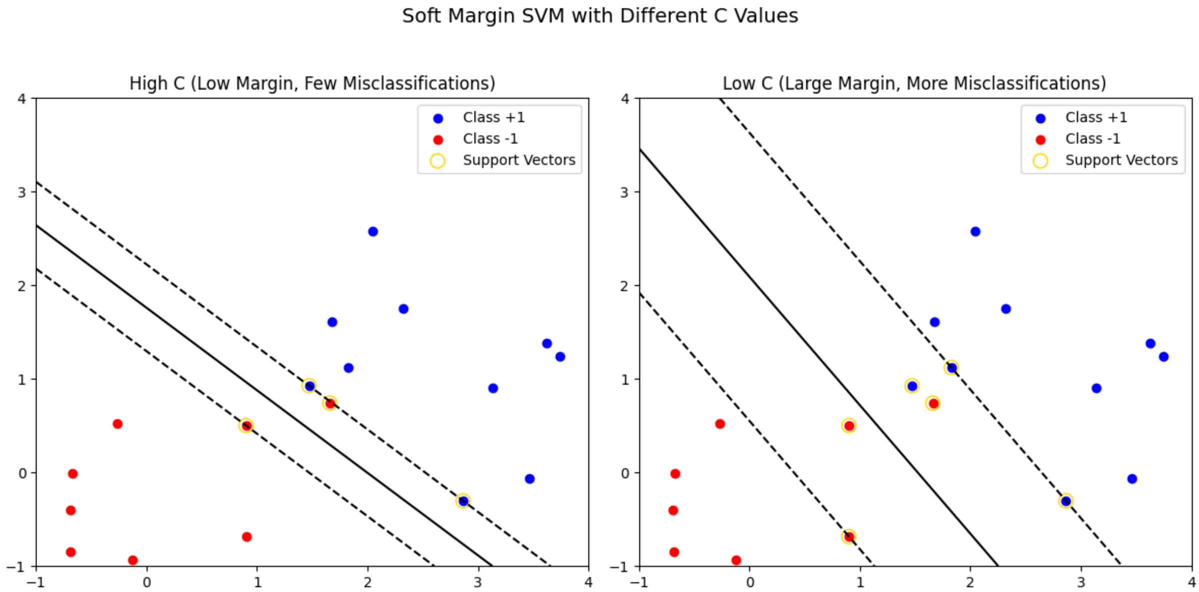

Figure 3.2: Effect of C on SVM soft margin.

3.3.3. Geometric Interpretation

In a soft margin SVM:

- The hyperplane still separates most of the data points correctly.
- Points that lie **inside the margin** or are **misclassified** contribute to the slack variables ξ_i.
- The support vectors continue to define the margin boundaries, but their role extends to accounting for margin violations.

3.3.4. Advantages of Soft Margins

The introduction of soft margins makes SVMs more flexible and applicable to noisy datasets:

- They allow SVM to handle overlapping classes where perfect separation is not possible.
- By tuning the parameter C, the model can achieve a balance between fitting the training data and generalizing to unseen data.

3.3.5. Dual Problem and Support Vectors

The function to be optimized in the soft margin Support Vector Machine is given by:

$$\text{Maximize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

To solve the SVM optimization problem, we introduce the concept of Lagrange multipliers α_i and β_i. The primal form of the optimization problem is expressed as:

$$L_p = \frac{1}{2}\|w\|^2 - C\sum_{i=1}^{I}\xi_i - \sum_{i=1}^{I}\alpha_i\left[y_i(w^T x_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{I}\beta_i\xi_i,$$

where:

- α_i are Lagrange multipliers associated with each constraint,
- C is the regularization parameter controlling the trade-off between margin width and misclassification,
- ξ_i are the slack variables accounting for margin violations.

By taking the gradient of L_p with respect to w and b and equating to zero, we obtain the following conditions:

$$\frac{\delta L_p}{\delta b} = 0 \implies \sum_{i=1}^{I}\alpha_i y_i = 0,$$

$$\frac{\delta L_p}{\delta w} = 0 \implies w = \sum_{i=1}^{I}\alpha_i y_i x_i.$$

Replacing these results into the primal form, we derive the **dual problem**:

$$L_D = \sum_{i=1}^{I}\alpha_i - \frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{I}\alpha_i\alpha_j y_i y_j x_i^T x_j.$$

This dual formulation must be maximized with respect to α_i under the following constraints:

$$\sum_{i=1}^{I}\alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0.$$

Types of Support Vectors

In the solution of the dual problem, the Lagrange multipliers α_i help identify the **support vectors**, which are critical points for defining the decision boundary:

- If $0 < \alpha_i < C$: The corresponding data point lies exactly on the margin, satisfying $y_i(w^T x_i + b) = 1$ with $\xi_i = 0$.
- If $\alpha_i = C$: The corresponding data point lies inside the margin or is misclassified. In this case, $\xi_i > 0$, and the condition $y_i(w^T x_i + b) = 1 - \xi_i$ holds.

Interpretation of Support Vectors The support vectors are the data points that define the margin and determine the optimal hyperplane. Points with $\alpha_i > 0$ contribute directly to the solution, while all other points ($\alpha_i = 0$) have no impact on the decision boundary. This sparsity property makes SVM computationally efficient.

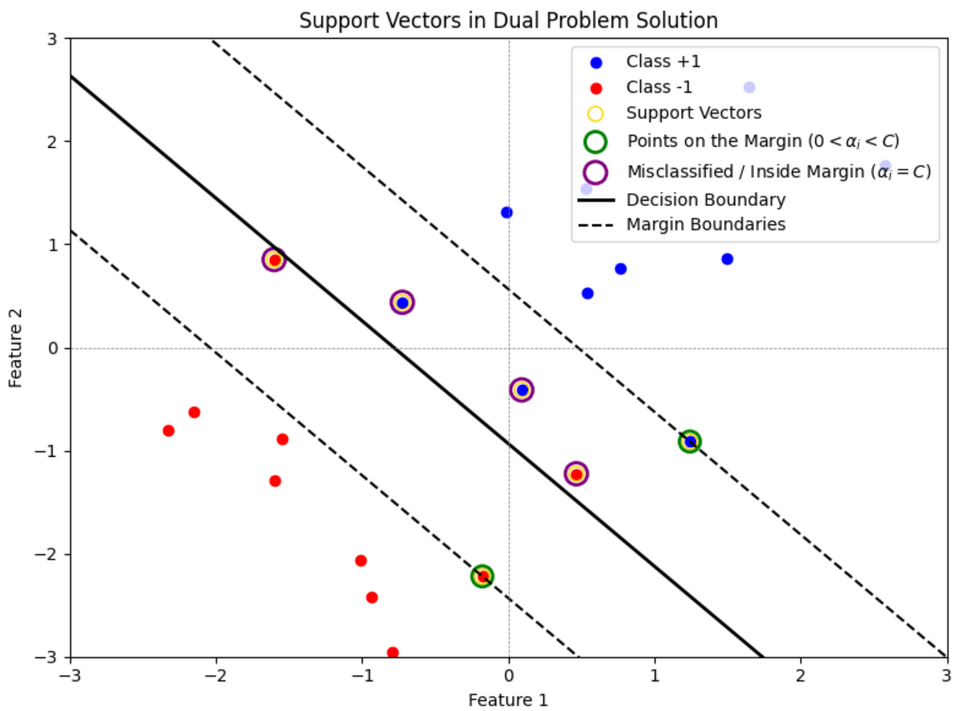

Figure 3.3: Illustration of the dual problem solution and support vector types. Points on the margin satisfy $0 < \alpha_i < C$, while points inside the margin or misclassified satisfy $\alpha_i = C$.

3.4. Non-Linear SVM Techniques

When the data distribution is so complex that even soft margins cannot separate the classes, Support Vector Machines (SVMs) employ **kernel methods** to handle non-linear problems. The key idea is to map the data into a higher-dimensional space where a linear hyperplane can separate the classes effectively.

3.4.1. The Kernel Trick

Instead of explicitly transforming the data into a higher-dimensional space, SVMs use the **kernel trick**, which allows the computation of the dot product in this space without performing the transformation directly. This reduces computational complexity while maintaining effectiveness.

Mathematically, the kernel function $K(x_i, x_j)$ replaces the dot product in the higher-dimensional space:

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$

where $\phi(x)$ is the mapping function to the higher-dimensional space. The SVM optimization problem is then solved using the kernel function, enabling efficient handling of non-linear problems.

3.4.2. Common Kernel Functions

Depending on the structure of the data, different kernel functions can be chosen:

- **Linear Kernel:** Suitable for linearly separable problems:

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

- **Polynomial Kernel:** Effective for polynomial relationships between features. It introduces parameters:

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d,$$

  where γ controls the influence of each feature, r is a constant, and d is the polynomial degree.

- **Radial Basis Function (RBF):** Commonly used for complex decision boundaries. It measures similarity between points:

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

  where $\gamma > 0$ controls the range of influence of each sample.

- **Sigmoid Kernel:** Inspired by neural networks and useful for specific data distributions:

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r).$$

3.4.3. Advantages of Kernel Methods

Kernel methods make SVMs highly versatile and capable of addressing a wide range of classification problems:

- They enable SVMs to handle non-linear decision boundaries without explicitly increasing the feature space dimensionality.
- By choosing an appropriate kernel function, SVMs can adapt to the underlying structure of the data.
- Kernel methods allow SVMs to efficiently classify data in high-dimensional spaces, making them robust for problems like image and text classification.

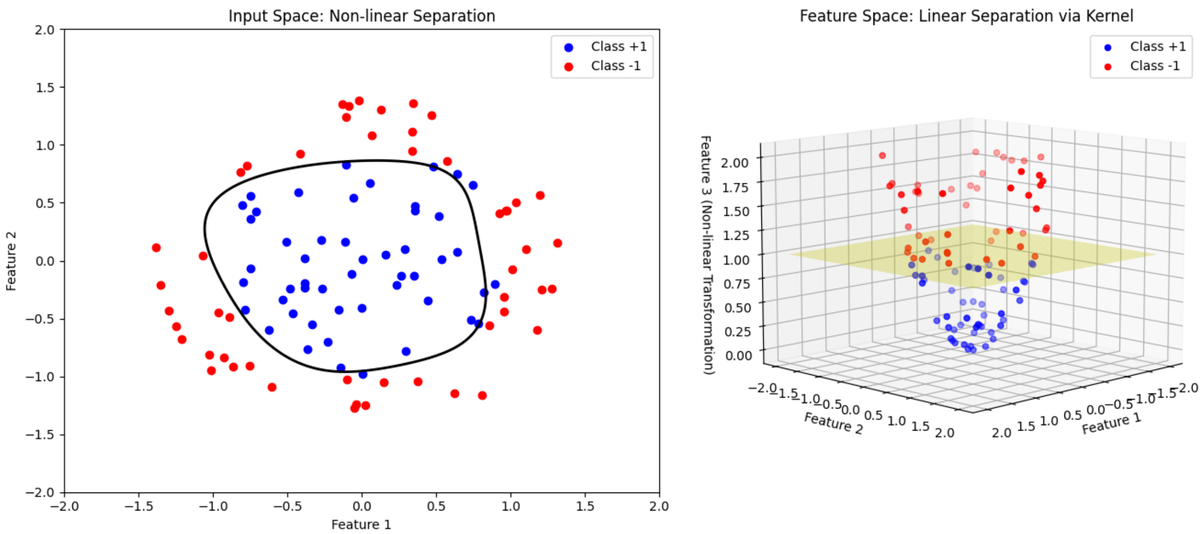

Figure 3.4: Illustration of the **Kernel Method** in SVM using RBF kernel. The left plot shows the *input space*, where the data is *non-linearly separable*. The right plot demonstrates how the *kernel trick* maps the data into a higher-dimensional *feature space*, where the classes become linearly separable. The yellow plane represents the hyperplane separating the two classes in the transformed space.

3.4.4. Challenges of Kernel Methods

Despite their flexibility, kernel methods come with certain challenges:

- **Choice of Kernel:** Selecting the right kernel function and its parameters (γ, r, d) often requires domain knowledge and experimentation.
- **Computational Cost:** For very large datasets, the computation of kernel matrices can become expensive in terms of both memory and time.
- **Overfitting Risk:** Using overly complex kernels may lead to overfitting, especially with small datasets.

4 | SVM for Regression

4.1. Differences Between Classification and Regression

In classification tasks, the objective is to assign a discrete label to an observation, typically representing membership in a specific category. In contrast, regression aims to predict a continuous output value based on input features. In Support Vector Machines (SVMs), the distinction manifests in how data is handled relative to the margin:

- **Classification:** The data points are placed outside the margin, with the goal of minimizing classification errors and maximizing the separation margin between classes.
- **Regression:** The data points are allowed to lie within the margin, and the objective is to minimize the deviation from a predicted value, often constrained by a tolerance parameter ϵ.

As seen in Figure 3.3, classification SVMs aim to find a decision boundary with the maximum margin between classes, while regression SVMs seek to fit a line within an ϵ-tube that minimizes deviations.

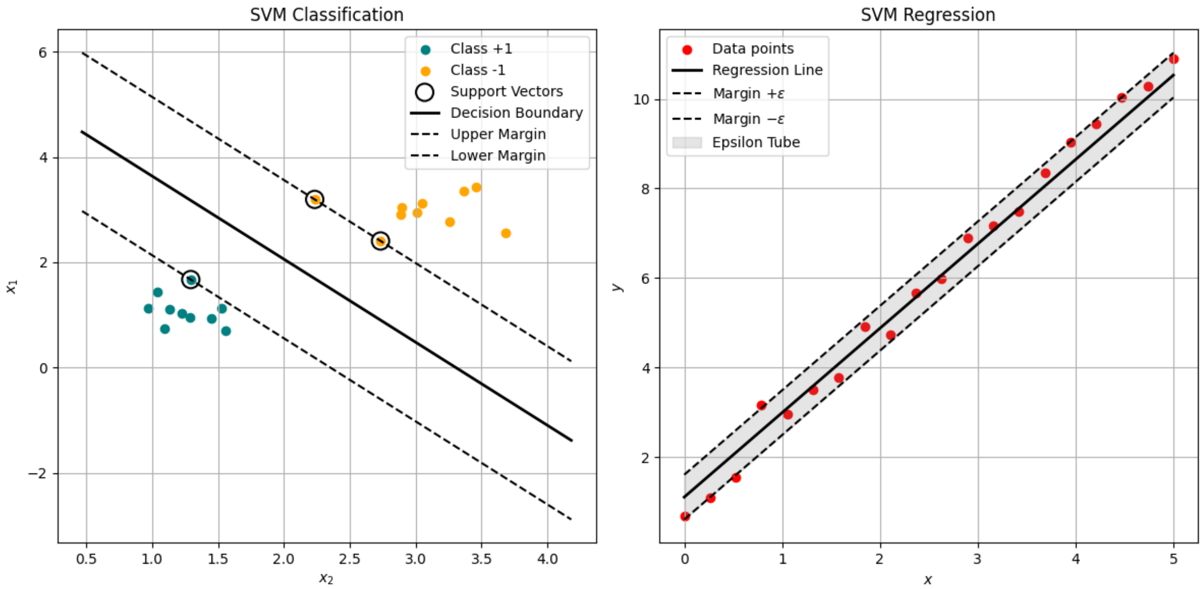

Figure 4.1: Comparison between SVM for classification (left) and regression (right).

4.2. Approaches to Regression

Support Vector Regression (SVR)[5] involves finding a function $f(x) = w^T x + b$ that predicts output values while balancing model complexity and prediction accuracy. This balance is achieved by solving the optimization problem:

$$\min \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*)$$

subject to:

$$y_i - (w^T x_i + b) \leq \epsilon + \xi_i,$$
$$(w^T x_i + b) - y_i \leq \epsilon + \xi_i^*,$$
$$\xi_i, \xi_i^* \geq 0,$$

where ξ_i, ξ_i^* are slack variables accounting for deviations beyond ϵ, and C is a regularization parameter controlling the trade-off between margin width and tolerance for prediction error.

4.3. Key Regression Formulas

The final regression function in SVR can be expressed as:

- **Linear Kernel:**

$$y(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b$$

- **Non-Linear Kernel:**

$$y(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \langle \phi(x_i), \phi(x) \rangle + b$$

- **General Kernel:**

$$y(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) K(x_i, x) + b$$

Here, $K(x_i, x)$ represents the kernel function that implicitly maps input data to a higher-dimensional space for linear separation.

5 | Applications of SVM in Real-World Problems

Support Vector Machines (SVMs) have proven to be versatile and powerful tools in a wide variety of domains. Their robustness to high-dimensional data, effective generalization, and ability to handle complex boundaries make them a preferred choice in many real-world applications. Below, we explore some key areas where SVMs have been successfully applied.

Image Processing

In computer vision, SVMs are widely applied for:

- **Object Detection:** Detecting faces, pedestrians, and other objects in images.
- **Image Classification:** Categorizing images into predefined categories based on their features.

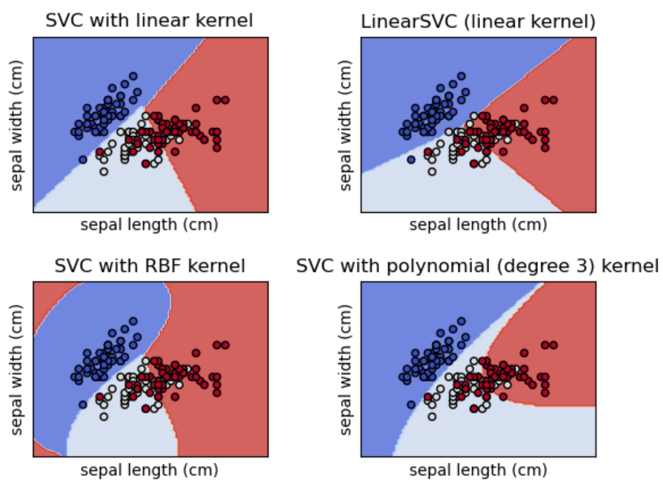

Figure 5.1: Comparison of SVM classifiers on the Iris dataset using different kernels. The decision boundaries vary based on the kernel function, demonstrating the flexibility of SVMs for non-linear data. Source: [6]

Bioinformatics

SVMs are extensively used in bioinformatics due to their ability to handle high-dimensional data with small sample sizes. Applications include:

- **Gene Classification:** SVMs are used to classify gene expression data for cancer diagnosis and prognosis.
- **Protein Structure Prediction:** SVMs aid in predicting secondary and tertiary structures of proteins.

Text and Sentiment Analysis

Text mining benefits greatly from SVMs, particularly for:

- **Spam Detection:** Classifying emails as spam or not spam.
- **Sentiment Analysis:** Determining the sentiment (positive, neutral, or negative) of reviews or social media posts.

Financial Applications

In finance, SVMs are used for:

- **Fraud Detection:** Identifying fraudulent transactions based on historical patterns.
- **Stock Market Prediction:** Predicting market trends and asset prices.

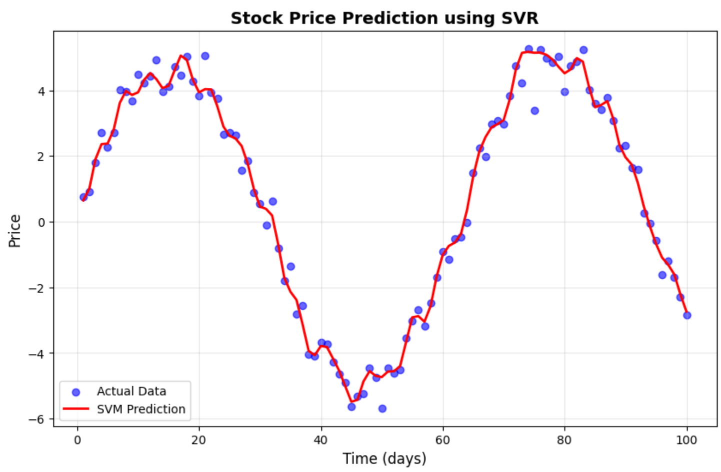

Figure 5.2: Stock Price Prediction using SVR.

Image Generation Note

Except for two images, which are cited accordingly, all figures and visualizations presented in this document were generated by the author using Python. The code utilized libraries such as `matplotlib`, `scikit-learn`, and `numpy` to ensure accurate representation of the concepts discussed.

Bibliography

[1] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[2] Munir Ahmad, Shabib Aftab, and Iftikhar Ali. Sentiment analysis of tweets using svm. *Int. J. Comput. Appl*, 177(5):25–29, 2017.

[3] Raquel Rodríguez-Pérez and Jürgen Bajorath. Evolution of support vector machine and regression modeling in chemoinformatics and drug discovery. *Journal of Computer-Aided Molecular Design*, 36(5):355–362, 2022.

[4] Wikimedia Commons contributors. SVM Margin Image. `https://commons.wikimedia.org/wiki/File:SVM_margin.png`, 2023. Licensed under CC BY-SA 4.0.

[5] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

[6] Scikit learn developers. Plot different svm classifiers in the iris dataset, 2024. Accessed: 2024-12-16.