University of Hertfordshire
M.Sc. Data Science with Placement Year
Machine Learning and Neural Networks

Fine-Tuning Support Vector Machines - A Guide to Hyperparameters and Feature Scaling

By Okeogheneomaero Evru (22030848)

December 2024

Github Link: https://github.com/maeintech/Fine-Tuning-SVMs

Imagine you are trying to build a machine learning model to classify spam emails. At first, simpler algorithms might suffice, but as complexity grows, you might decide to turn to Support Vector Machines (SVMs)—known for their precision in drawing the optimal boundary between classes. However, SVMs don't perform optimally straight out of the box. Their performance hinges on several interconnected factors that must be carefully balanced to suit the dataset and task at hand. Fine-tuning these aspects is a process of thoughtful adjustment and iteration, progressively refining the model to better capture the patterns in the data.

This tutorial will focus on two key elements of this process: hyperparameter tuning and feature scaling. These areas are crucial for enhancing the performance of SVM models, especially when dealing with complex datasets. By exploring the details of how parameters like the regularization term, C and kernel coefficient, gamma influence the decision boundary, along with the importance of preprocessing techniques like feature scaling, this tutorial equips the reader with the tools to optimize their own SVM models.

The focus on hyperparameter tuning and feature scaling helps readers understand how to make precise adjustments to the SVM algorithm, ensuring it is tailored for specific tasks such as classification of complex datasets or image recognition. Mastering these aspects will enable practitioners to achieve better model accuracy and generalization, ultimately improving the effectiveness of their machine learning solutions.

What Exactly are Support Vector Machines, Anyway?

Support Vector Machines (SVMs) are among the most versatile and powerful algorithms in machine learning, widely used for both classification and regression tasks. Their fundamental principle is simple yet highly effective: find the optimal boundary that separates classes in the dataset. This boundary, called the hyperplane, is positioned to maximise the margin between the closest points of each class, known as support vectors. This capability allows SVMs to excel in tasks requiring high precision, such as text classification, bioinformatics, and image recognition [1].

Figure 1 below illustrates the concept of margins in Support Vector Machines (SVM). A good margin maximizes the distance between the support vectors of two classes and the decision boundary, ensuring better generalization. Conversely, a bad margin occurs when the support vectors are too close to the boundary, leading to a smaller margin and potentially poorer classification performance. The goal of SVM is to achieve the largest possible margin for optimal separation of classes.

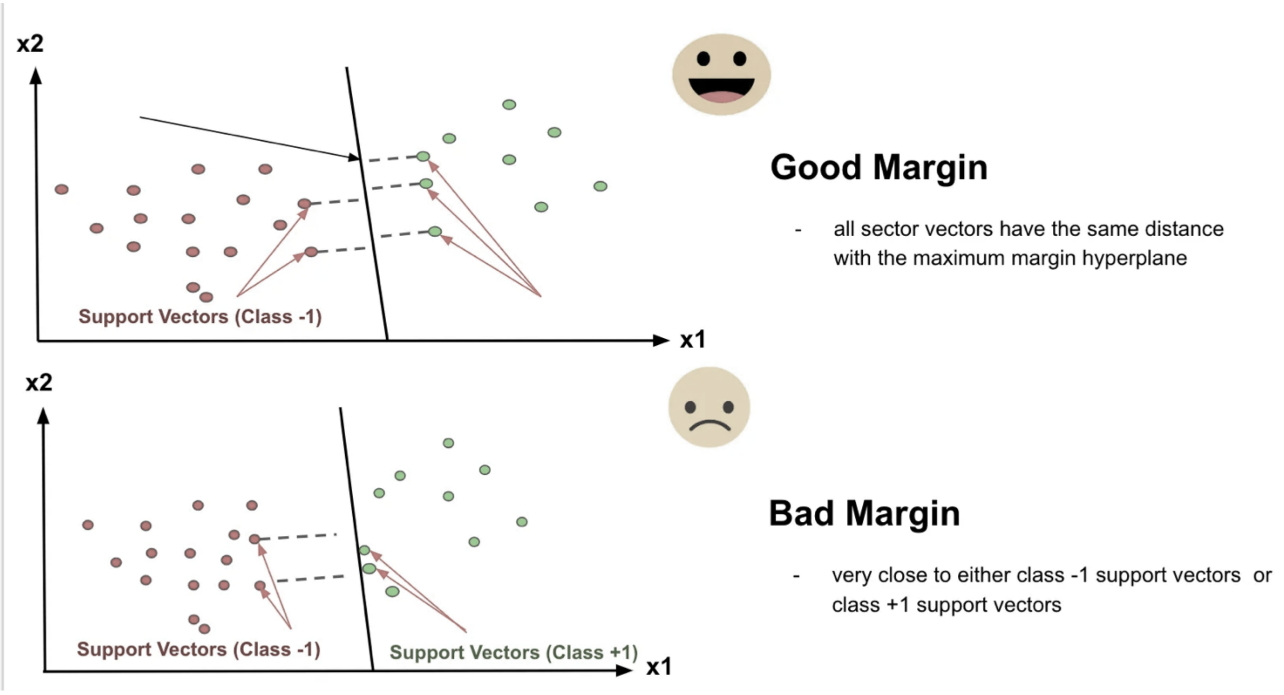
x2
x1
Support Vectors (Class -1)
Good Margin
- all sector vectors have the same distance with the maximum margin hyperplane
x2
x1
Support Vectors (Class -1)
Support Vectors (Class +1)
Bad Margin
- very close to either class -1 support vectors or class +1 support vectors

Figure 1: Good vs Bad Margins in SVM: Demonstrating the effect of margin size and support vectors on classification performance. [2]

Factors affecting SVM Performance

The performance of an SVM model depends on several interacting factors that influence its ability to generalise effectively. The table below summarizes the effects of key factors on SVM model performance:

Factor	Definition	Impact on Model Performance
Kernel Choice	Determines how data is mapped into higher-dimensional space.	• Affects the ability to separate non-linear data. • Improper choice can underfit or overfit the data.
Regularization (C)	Controls the trade-off between a wide margin and classification accuracy.	• **Low C**: Wide margin, more generalization, tolerates misclassifications. • **High C**: Narrow margin, higher accuracy, but risk of overfitting.
Kernel Coefficient (gamma)	Determines the influence of individual data points in the RBF kernel.	• **Low gamma**: Smoother boundary, risk of underfitting. • **High gamma**: Complex boundary, risk of overfitting.
Feature Scaling	Ensures all features contribute equally by standardizing their ranges.	• Prevents features with large ranges from dominating distance-based calculations. • Improves model accuracy.
Class Imbalance	Uneven distribution of target classes in the dataset.	• Leads to biased decision boundaries favoring the majority class. • Requires techniques like class weights.

Table 1: Factors affecting SVM performance

While all these factors are crucial, this tutorial will focus specifically on **hyperparameters** and **feature scaling** to explore how they impact the performance of an SVM algorithm. By examining these two areas, we will better understand their roles in shaping the model's accuracy and effectiveness in real-world applications.

Why Hyperparameters and Feature Scaling Matter

Hyperparameters are like the settings on a musical instrument, such as a guitar. Just as you adjust the tension of the strings, the volume, or the tone to create the perfect sound, hyperparameters like the kernel, C, gamma, and degree let you fine-tune how an SVM learns from the data. Each setting plays a role in shaping the model's performance, much like how small changes to the instrument's settings can drastically alter the music it produces. When done right, these adjustments make the SVM "sing" by achieving the best possible predictions on your data. On a more granular level, hyperparameters like C and gamma are important in shaping the decision boundary, directly impacting the model's ability to generalise across datasets. Improper tuning can lead to overfitting (when the model is too complex) or underfitting (when it is too simplistic). [3]

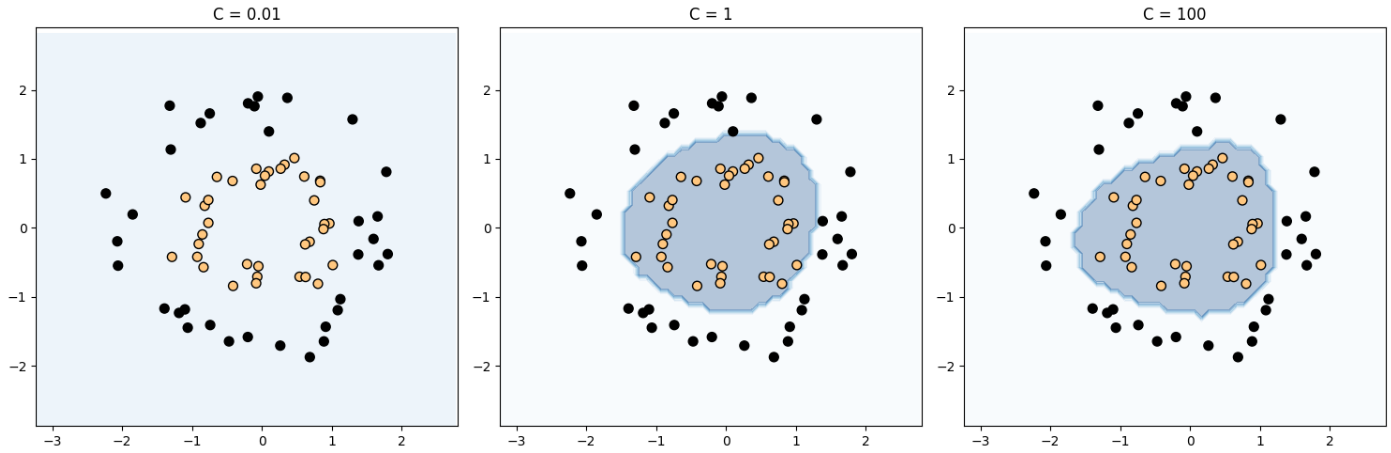

Figure 2: Impact of regularization parameter C on SVM decision boundaries: Low C (left) leads to underfitting, high C (right) leads to overfitting, and moderate C (center) results in a well-balanced model.

Similarly, feature scaling is like tuning a set of speakers to ensure consistent sound across different frequencies. Imagine you have a stereo system with multiple speakers: one for bass, one for mid-range, and one for treble. If one speaker is too loud compared to the others, the overall sound won't be balanced. To ensure that all frequencies contribute equally to the sound, you adjust the volume of each speaker to be at the same level. Without proper feature scaling during preprocessing, distance calculations can become skewed, causing the model to misinterpret the relationships between data points. In machine learning, features with large values (like income in thousands) can overwhelm smaller features (like age in years). Feature scaling adjusts the range of these features so that they all contribute equally to the model's performance, just like balancing the volume of each speaker for a well-rounded sound. [4]

How Do Hyperparameters work in SVM?

The decision boundary of an SVM model is the line (or hyperplane in higher dimensions) that separates different classes in the feature space. Hyperparameters like **regularization parameter (C), the kernel coefficient (gamma)** and the **kernel type** play a crucial role in defining this boundary:

- **C (Regularization Parameter)**: The parameter C controls the trade-off between maximizing the margin between classes and minimizing classification errors. It directly influences the complexity of the decision boundary that the SVM model learns. A small C value allows the model to tolerate more misclassifications by creating a wider margin. This results in a simpler decision boundary, which may be more generalizable to unseen data. However, the model might misclassify some data points in the training set, especially in noisy or complex datasets. On the other hand, a larger C value prioritizes classification accuracy, forcing the model to classify training data more correctly by reducing the margin. This results in a more complex decision boundary that might be too tightly fitted to the training data, leading to overfitting if the model becomes too sensitive to minor variations in the data.

- **gamma (Kernel Coefficient)**: The kernel coefficient (gamma) is another crucial hyperparameter in SVM, particularly when using the Radial Basis Function (RBF) kernel. Gamma controls the influence of individual data points on the decision boundary and determines the smoothness of the resulting model. A small gamma means that each data point has a larger influence, leading to a smoother decision boundary. The model might be too simple and not capture the complexities of the data, leading to underfitting. Conversely, a larger gamma value means that each data point has a smaller influence, creating a more complex decision boundary that closely follows the training data. While this can improve performance on the training set, it may cause the model to overfit by being overly sensitive to the noise and small variations in the data

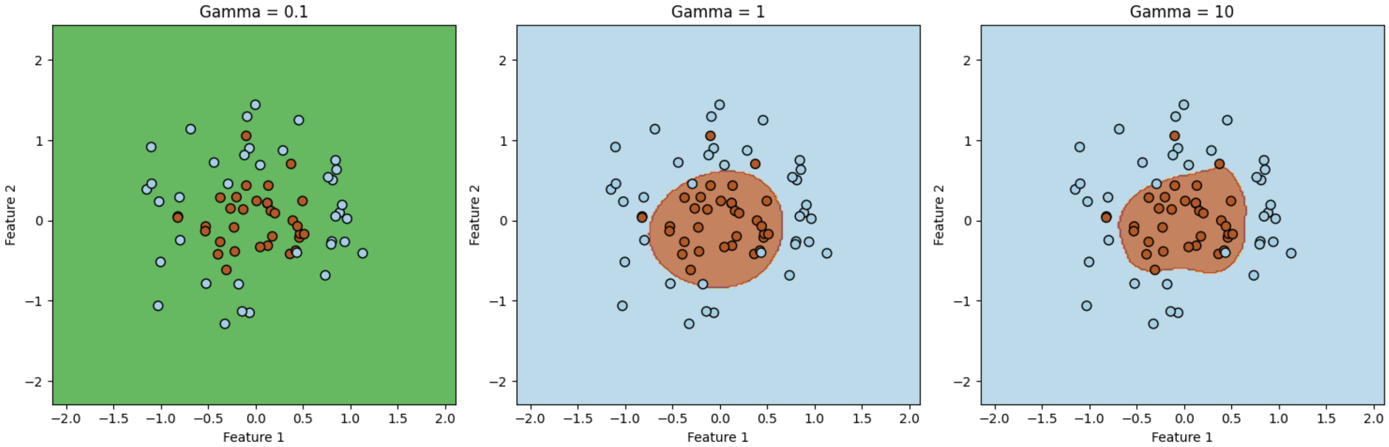

Figure 3: Decision boundaries for an SVM using RBF kernel at different gamma values (0.1, 1 and 10). As the gamma value increases, the decision boundary becomes more complex and tightly fits the training data, highlighting the effect of gamma on model flexibility and its potential to cause overfitting or underfitting.

- **Kernel Choice**: The kernel function is a critical component of SVMs because it transforms data into a higher-dimensional space where a linear separation is possible. The choice of kernel, along with the tuning of C and gamma, plays a fundamental role in determining the SVM's performance.
    - **Linear Kernel:** This is the simplest kernel, suitable when the data is linearly separable. It doesn't require the adjustment of gamma (since it's essentially a dot product in the original feature space) and is often preferred when the data is straightforward and separable
    - **Polynomial Kernel:** The polynomial kernel maps the data into higher-dimensional space using polynomial functions. It's useful for capturing relationships between features, but it can be sensitive to the degree of the polynomial. The kernel's complexity increases with the degree parameter, which can interact with C and gamma to control overfitting
    - **RBF (Radial Basis Function) Kernel:** The RBF kernel is popular for non-linear data. It transforms data using a Gaussian function, enabling SVM to separate classes in cases where a linear boundary is insufficient. Both C and gamma play significant roles in RBF kernels, where gamma controls the smoothness of the decision boundary and C controls the trade-off between margin size and classification error

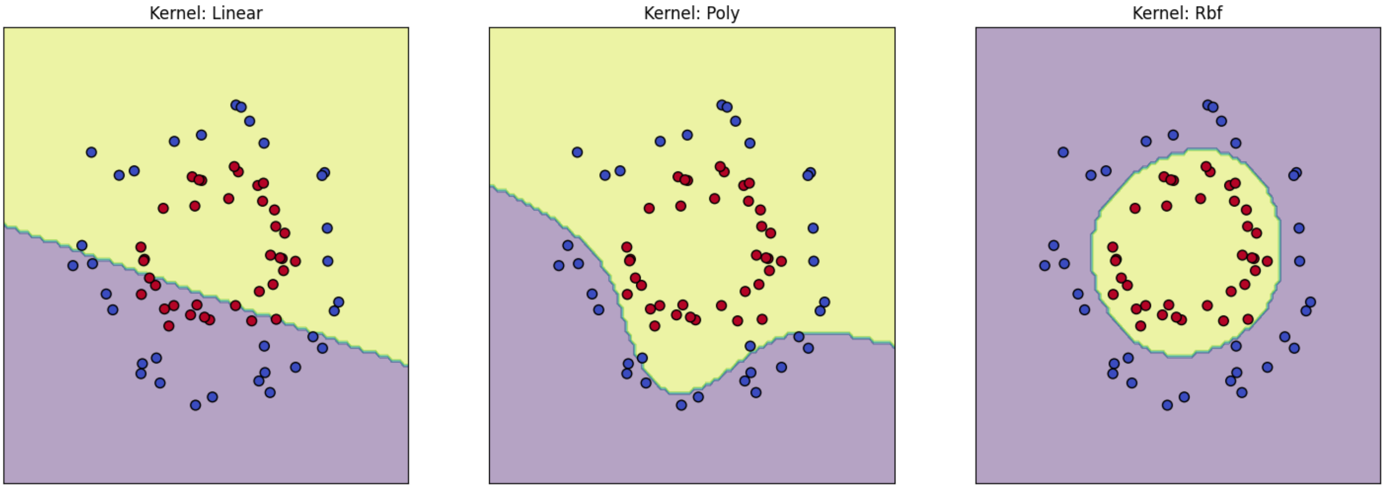

Figure 4: Decision boundaries for linear, polynomial, and RBF kernels applied to the same dataset. The plot illustrates how different kernel choices impact the model's ability to separate data, with the RBF kernel providing the most flexibility for non-linear datasets

How Feature Scaling Works in SVM

Feature scaling is an essential preprocessing step in many machine learning algorithms, including Support Vector Machines (SVM). Without scaling, features with larger ranges can dominate the model's performance, leading to suboptimal decision boundaries. Feature scaling ensures all features contribute equally, allowing the model to generalize better [5].

It involves transforming the feature values to a common scale without distorting differences in the ranges of values. Two commonly used methods of feature scaling are Standardization and Normalization

- **Standardization (Z-score scaling):** This involves transforming the features such that they have a mean of 0 and a standard deviation of 1. This is done by subtracting the mean of the feature and dividing by the standard deviation.

  $$z = \frac{x - \mu}{\sigma}$$

  where x is the feature value, μ is the mean of the feature, and σ is the standard deviation.

  Standardization is commonly used when the data has outliers or when you need to preserve the variance in the data (as is the case with SVMs).

- **Normalization (Min-Max scaling):** Normalization rescales the data to a fixed range, usually between 0 and 1. The feature values are rescaled by subtracting the minimum value and dividing by the range (the difference between the maximum and minimum values).

  $$x' = \frac{x - \min{(x)}}{\max(x) - \min{(x)}}$$

  where x is the feature value, and min(x) and max(x) are the minimum and maximum values of the feature.

  Normalization is typically used when the data is not normally distributed or when you need the features to be bounded within a certain range.

How does feature scaling impact SVM models in particular, you may ask? SVMs rely on the concept of distance between data points to find the optimal hyperplane. The algorithm works by maximizing the margin between the support vectors (the points closest to the hyperplane). If one feature has a much larger range than another, it will dominate the distance calculations, potentially skewing the model.

- **Distance Metric Sensitivity:**
  SVM uses distance metrics such as Euclidean distance to calculate the margin. Features with larger numerical values will disproportionately affect the distance calculation. For example, a feature representing income (in thousands of dollars) could dominate the feature representing age (in years), leading to poor model performance.

- **Kernel Function Sensitivity:**
  Kernel functions in SVM (like RBF) are sensitive to the scale of the input data. If the features are on different scales, the kernel may not behave as expected, leading to poor performance. For instance, in an RBF kernel, a small variance in one feature could cause the kernel to overreact, resulting in overfitting.

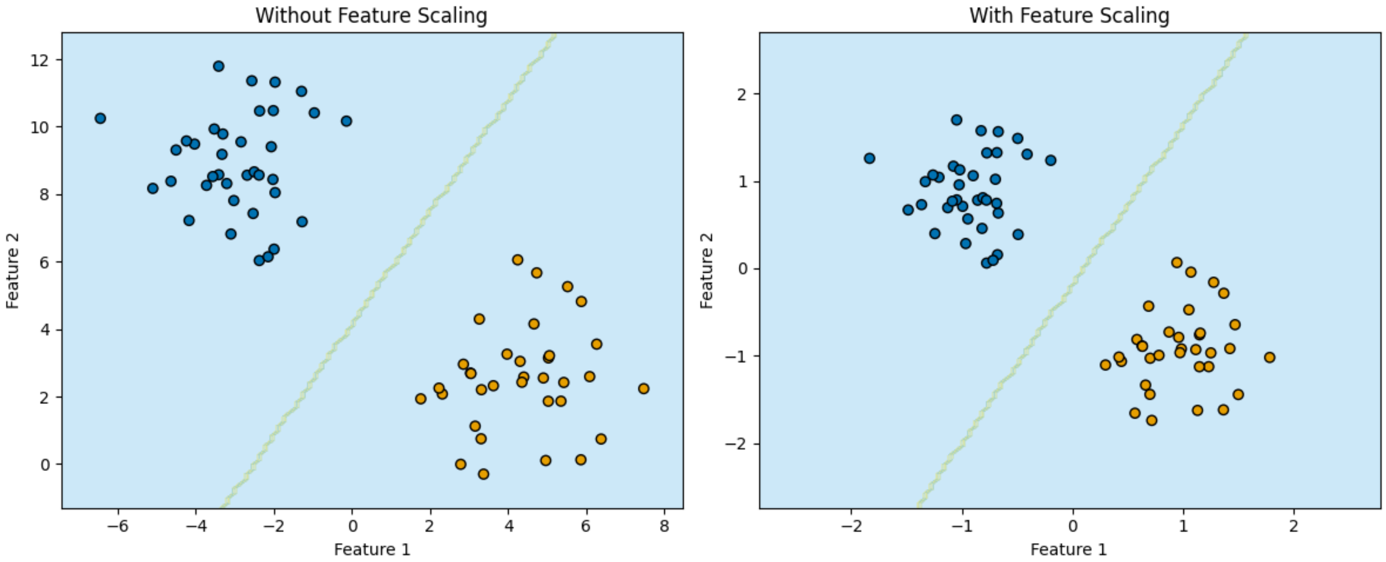

Figure 5: Effect of feature scaling on SVM decision boundaries: The left plot shows a skewed decision boundary without scaling, caused by the dominance of one feature. The right plot illustrates a balanced decision boundary after scaling, highlighting the importance of preprocessing steps like standardization to improve model performance

Practical Implementation with Python

We will now use Python to demonstrate how to generate a synthetic dataset, train a Support Vector Machine (SVM) classifier, visualize decision boundaries, and explore the impact of hyperparameters and feature scaling.

Step 1: We start by creating a synthetic dataset for classification tasks using make_classification from sklearn.datasets. This function allows us to generate a dataset with specified characteristics.

```
from sklearn.datasets import make_classification

# Generate synthetic dataset
X, y = make_classification(
   n_samples=200, n_features=2, n_informative=2, n_redundant=0,
   n_clusters_per_class=1, class_sep=1.2, random_state=42
)
```

In this case, we create 200 samples with 2 features, 2 informative features, and a class separation of 1.2 for better distinction between the classes.

Step 2: To evaluate model performance, the dataset is split into training and testing sets using train_test_split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,  y,  test_size=0.3, random_state = 42)
```

We allocate 30% of the data for testing and use the rest for training the model.

Step 3: A crucial part of understanding SVMs is visualizing how they classify data. We define a helper function to plot decision boundaries created by an SVM model.

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

def plot_decision_boundary(X, y, clf, title, ax):
   x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
   y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
   xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                  np.arange(y_min, y_max, 0.1))

   Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
   Z = Z.reshape(xx.shape)

   ax.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.viridis)
   scatter = ax.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', cmap=plt.cm.viridis, marker='o')
   ax.set_title(title)
   legend1 = ax.legend(*scatter.legend_elements(), title="Classes")
   ax.add_artist(legend1)
```

This function takes the model (clf) and the data (X, y) and generates a plot showing the decision boundary for classifying the data. It also shows the points, color-coded by their class labels.

Step 4: The regularization parameter C influences how strict the SVM is about classifying points correctly. Let's visualize the decision boundaries for different values of C:

```
from sklearn.svm import SVC

fig, axes = plt.subplots(2, 3, figsize=(15, 10))
for i, C inenumerate([0.1, 1, 10]):
   svc = SVC(C=C, kernel='rbf', gamma=1)
   svc.fit(X_train, y_train)
   plot_decision_boundary(X_train, y_train, svc, f"C={C}", axes[0, i])
plt.tight_layout()
plt.show()
```

This code fits the SVM model for different values of C (0.1, 1, and 10) and visualizes how the decision boundary changes. A lower C allows more misclassifications (looser margin), while a higher C enforces fewer misclassifications (stricter margin). The resulting plots can be found in the figure below showing the effect different values of C have on the SVMs ability to cluster.

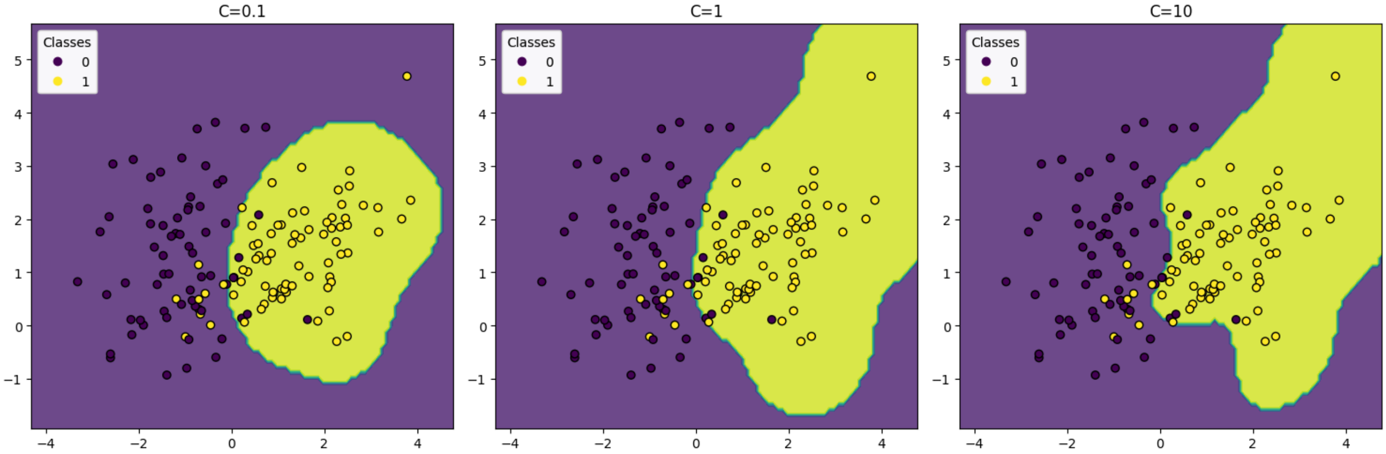

Figure 6: Impact of regularization parameter (C) on SVM decision boundaries. From left to right: C = 0.1 (underfitting), C = 1 (balanced), C = 10 (overfitting)

Step 5: The gamma parameter controls the curvature of the decision boundary. We explore its effect by training the model with different values of gamma:

```
for i, gamma in enumerate([0.1, 1, 10]):
    svc = SVC(C=1, kernel='rbf', gamma=gamma)
    svc.fit(X_train, y_train)
    plot_decision_boundary(X_train, y_train, svc, f"Gamma={gamma}", axes[1, i])

plt.tight_layout()
plt.show()
```

This experiment shows how changing gamma affects the flexibility of the decision boundary. A higher gamma leads to a more complex decision surface. The resulting plots can be found in the figure below showing the effect different values of gamma have on the SVMs create a smooth decision boundary.

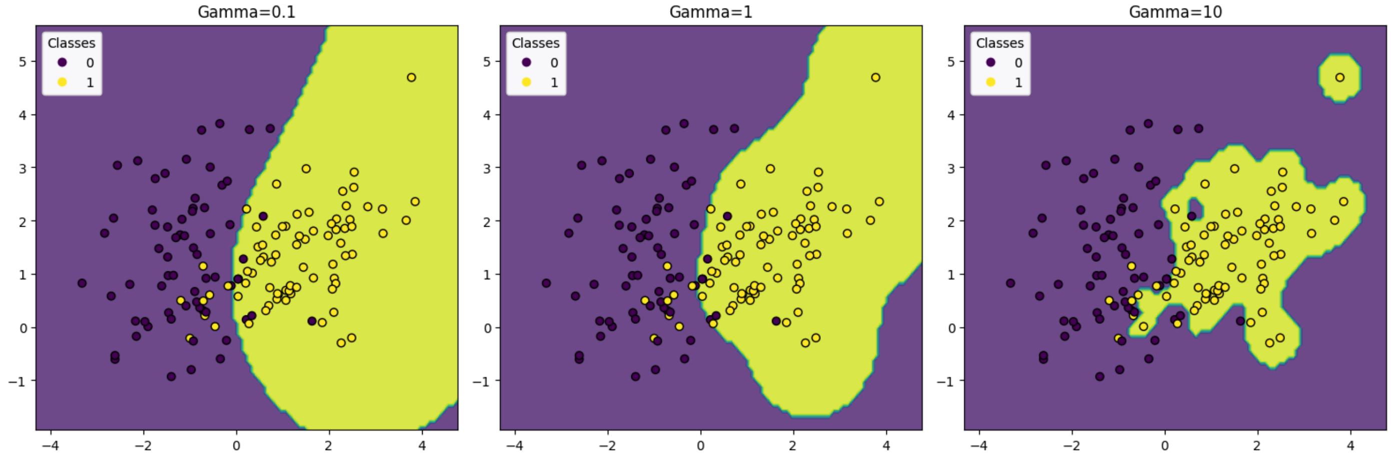

Figure 7: Effect of kernel coefficient (gamma) on SVM decision boundaries. From left to right: gamma=0.1, 1 and 10

Step 6: SVMs are sensitive to the scale of features. We demonstrate the importance of feature scaling by training the model on unscaled data and scaled data:

```
from sklearn.preprocessing import StandardScaler

# Introduce unscaled data
X_unscaled = X_train.copy()
X_unscaled[:, 0] *= 100  # Disproportionately scale one feature

svc_no_scaling = SVC(C=1, kernel='rbf', gamma=1)
svc_with_scaling = SVC(C=1, kernel='rbf', gamma=1)

# Without scaling
svc_no_scaling.fit(X_unscaled, y_train)

# With scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_unscaled)
svc_with_scaling.fit(X_scaled, y_train)

# Compare decision boundaries
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
plot_decision_boundary(X_unscaled, y_train, svc_no_scaling, "Without Scaling", ax[0])
plot_decision_boundary(X_scaled, y_train, svc_with_scaling, "With Scaling", ax[1])
plt.show()
```

In this case, we artificially scale one feature to emphasize the impact of scaling. As you can see, SVM performance improves when features are scaled, as the classifier is no longer biased by the differing magnitudes of the features.

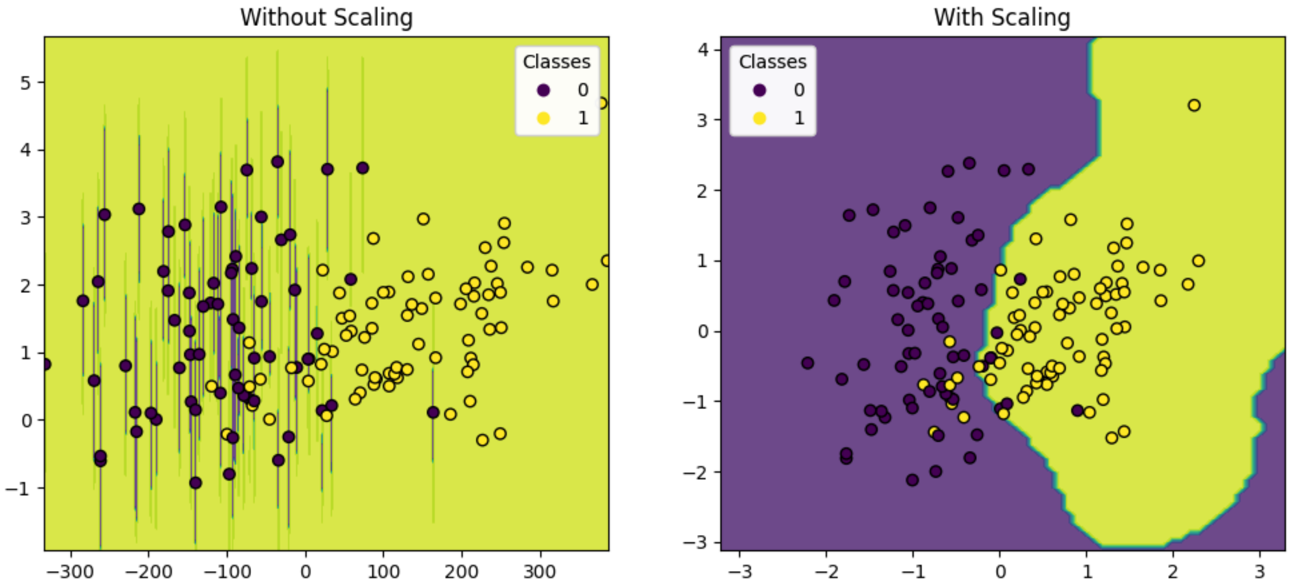

Figure 8: Effect of feature scaling on SVM decision boundaries. Left: Unscaled data, showing a biased decision boundary. Right: Scaled data, with a balanced decision boundary after feature scaling.

Key Takeaways

Support Vector Machines (SVMs) offer robust and versatile solutions for both classification and regression tasks, thanks to their ability to delineate complex decision boundaries with precision. However, achieving optimal performance with SVMs requires careful attention to hyperparameters and preprocessing techniques such as feature scaling.

In this tutorial, we explored the key hyperparameters—regularization parameter C, kernel coefficient (gamma), and kernel type—and explored their effects on the model's complexity and generalization. Through practical examples and visualizations, we demonstrated how adjusting these settings can fine-tune an SVM's decision boundaries for improved performance.

We also underscored the importance of feature scaling, an often-overlooked preprocessing step that ensures all features contribute equally to the model's computations. By comparing scaled and unscaled data, we illustrated how proper scaling can enhance the accuracy and reliability of an SVM classifier.

By mastering these concepts, practitioners can unlock the full potential of SVMs and apply them effectively to a wide range of data science challenges. As you experiment with these techniques, remember that the key to success lies in iterative exploration and thoughtful adjustment. Whether tackling spam classification, image recognition, or bioinformatics, the principles discussed here will serve as a foundation for building more accurate and reliable models.

Works Cited

[1] L. Diosan , A. Rogozan and J.-P. Pecuchet, “Improving classification performance of Support Vector Machine by genetically optimising kernel shape and hyper-parameters,” *Applied Intelligence,* vol. 36, no. 2, pp. 280 - 294, 2012.

[2] R. Sivakumar, “Demystifying Support Vector Machine from Scratch,” Medium, 8 March 2021. [Online]. Available: https://medium.com/srm-mic/demystifying-support-vector-machine-from-scratch-edaaaba4bda. [Accessed 3 December 2024].

[3] P. Shrestha, “Tuning SVM Hyperparameters: Making Your Classifier Shine Like a Pro!,” Medium, 9 September 2024. [Online]. Available: https://medium.com/@prayushshrestha89/tuning-svm-hyperparameters-making-your-classifier-shine-like-a-pro-8673639ddb16. [Accessed 3 December 2024].

[4] Scikit-learn, “Preprocessing Data,” [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html. [Accessed 5 December 2024].

[5] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning,* vol. 20, pp. 273-297, 1995.