Title

Comparison of Kernel Functions and Parameter Selection of SVM Classification Algorithms

Author

Pan, Linying

Publication Date

2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Comparison of Kernel Functions and Parameter Selection of SVM Classification Algorithms

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics and Data Science

by

Linying Pan

2023

ABSTRACT OF THE THESIS

Comparison of Kernel Functions and
Parameter Selection of SVM
Classification Algorithms

by

Linying Pan
Master of Applied Statistics and Data Science
University of California, Los Angeles, 2023
Professor Yingnian Wu, Chair

Support Vector Machine (SVM) is a reliable supervised learning model extensively utilized for classification and regression tasks, owing to its remarkable ability to achieve strong generalization performance. This study focuses on two key factors in the SVM model: the error penalty parameter C and the kernel function. The C parameter is used to balance the model's complexity and empirical risk, and its selection is crucial for SVM performance. A smaller C value may lead to underfitting, while a larger C can result in overfitting. Additionally, the choice of the kernel function also significantly impacts SVM performance. We will investigate the effects of different kernel functions and parameter settings in the classification task of the Iris dataset and visualize their impacts through a visual approach. The study's results indicate that, in most cases, the Gaussian kernel outperforms other kernel functions, exhibiting superior classification performance and generalization capability. Therefore, we opt for the Gaussian Radial Basis Function (RBF) kernel and conduct experiments to evaluate the influence of different parameter configurations on classification performance.

The thesis of Linying Pan is approved.

Michael Tsiang

Qing Zhou

Yingnian Wu, Committee Chair

University of California, Los Angeles

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The Support Vector Machine (SVM) is both a supervised learning model and a related learning algorithm. It is built on the foundation of statistical learning theory, including Vapnik Chervonenk (VC) dimension theory and Structural Risk Minimization (SRM) principles. Namely, SVM constructs decision rules from a finite set of training samples, aiming to achieve low errors when applied to independent test sets [1]. In addition, It finds extensive applications in tasks related to classification and regression analysis. As a novel machine learning algorithm, SVM is renowned for its remarkable generalization performance, thus making it an essential tool in machine learning. Nevertheless, SVM's classification performance is also influenced by various factors, primarily the choice of the error penalty parameter C and the form and parameters of the kernel function.

The error penalty parameter C in the SVM balances the misclassification rate and algorithm complexity. It fine-tunes the machine's confidence level and empirical risk ratio within a defined feature subspace, aiming for optimal generalization capability [2]. When selecting the error penalty parameter C, one must consider the fundamental nature of the problem and the presence of noise points in the data. When C takes on a large value, even approaching infinity, SVM becomes very strict in classifying the training data. In other words, it is required that all training samples must be classified entirely without error. However, this results in higher model complexity, thus potentially overfitting the training data, and consequently, decreasing generalization performance. Conversely, as C reduces, a wide margin decision boundary is chosen, effectively disregarding whether the training

samples are correctly classified, which can lead to underfitting issues.

So far, the kernel functions involved in research primarily encompassed linear kernel, Gaussian kernel, polynomial kernel, and sigmoid kernel. SVM kernel functions are introduced to overcome the problem of data linear inseparability. This is achieved by mapping the data to a high-dimensional space, thus addressing the issue of linear inseparability in the original area. It is expected that samples in the feature space exhibit linear separability, so the quality of the feature space is crucial for the performance of SVM. This can also be seen when constructing a well-performing support vector machine, the key lies in choosing the kernel function.

This article aims to investigate the influence of kernel functions and their associated parameters on the classification performance of SVM. Many scholars have conducted extensive experimental studies, and these experiments indicate that, in many cases, the Gaussian Radial Basis Function (RBF) kernel outperforms other kernel functions regarding classification effectiveness and generalization. Therefore, the parameter settings for the RBF kernel have always been one of the focal points of research. However, how to choose appropriate parameters is still a question worth exploring. In my experiments, I focused on two features within the Iris dataset and conducted extensive comparative analyses. The results verify the superiority of the Gaussian kernel function, which exhibited the shortest training time and the highest testing accuracy. Furthermore, my research delved into the impact of SVM parameters on performance. It was evident from the result that parameters C and gamma directly influenced feature classification. So, configuring the appropriate penalty factor and kernel parameters is equally crucial for optimizing the classification model. Then, to optimize these parameters, I also introduced the Tree-structured Parzen Estimator (TPE) method, leveraging Bayesian optimization tools. The application of TPE allowed us to identify and cross-validate the optimal hyperparameter combination, leading to a further enhancement in model performance.

CHAPTER 2

Methodology

2.1 The Basic Principles of SVM

SVM is a specialized classifier based on the principle of maximizing margins [3]. Simultaneously, it functions as a supervised learning algorithm primarily designed for tackling binary classification problems. As we all know, the fundamental principle of this model is to identify the optimal separation hyperplane within the feature space, aiming to maximize the margin between positive and negative samples in the training dataset. The overall data classification effect will improve as the distance between the two classification hyperplanes increases. With the introduction of kernel methods, SVM can also deal with non-linear problems. Therefore, the leading modeling forms of the SVM algorithm are divided into two types, namely linearly separable and linearly inseparable [4].

If the sample data can be linearly separated, the principle of SVM is shown in the following figure.

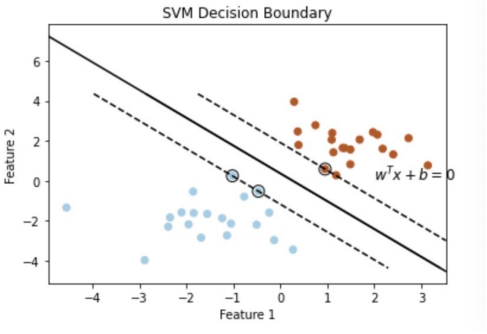

Figure 2.1: SVM Principle Illustration

2.2 Hyperplane and Margin

Given a set of data points that are divided into two different categories. In order to effectively separate these data points into these two categories, it is necessary to find a linear classifier. As shown in Figure 2.2, if we want to separate the two types of data points completely, there are many straight lines we can choose. Therefore, the focus of the study is how to find an optimal straight line. Let x represent the data points, and y represent their categories. For a linear classifier, the learning objective is to find a hyperplane in an n-dimensional data space. A hyperplane is a subspace or a particular plane in a higher-dimensional area that has one dimension less than the space it resides in. In this binary classification problem, the equation of the hyperplane can be represented as follows:

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

Here, "w" denotes the normal vector of the hyperplane, "x" represents the data points, and "b" corresponds the bias term.

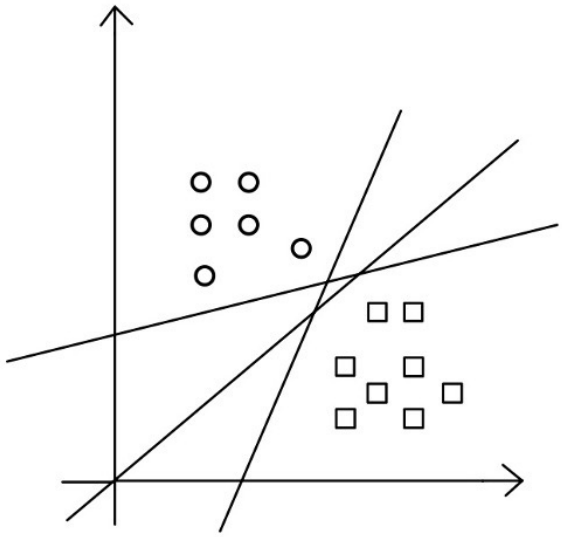

Figure 2.2: Linear Separability and Hard Margin Plot

Additionally, for the selection of the optimal hyperplane, the following criteria must be

met:

(1) Able to effectively separate two different categories.

(2) Have the largest margin, that is, have as much space as possible between the two categories.

(3) In the middle position of the margin, the distance from all support vectors is equal.

The SVM algorithm uses the size of the class interval to determine the optimal classifier, where the optimal classifier is the classifier with the largest classification interval. As shown in Figure 2.3, the figure shows the classification margin, which is the distance between the two extreme positions where the two solid lines are parallel to the decision surface. Once beyond these two positions, the sample points will face being misclassified. The optimal classifier can maximize the classification margin. Generally speaking, different decision hyperplanes will have different classification margins. To find the optimal decision surface in the SVM algorithm, you must first find the decision surface with the largest classification margin. It can be seen from Figure 2.3 that the size of the classification margin depends on the position of two straight lines parallel to the classification line. When a decision boundary is chosen, the support vectors can determine the positions of two straight lines parallel to the classification boundary.

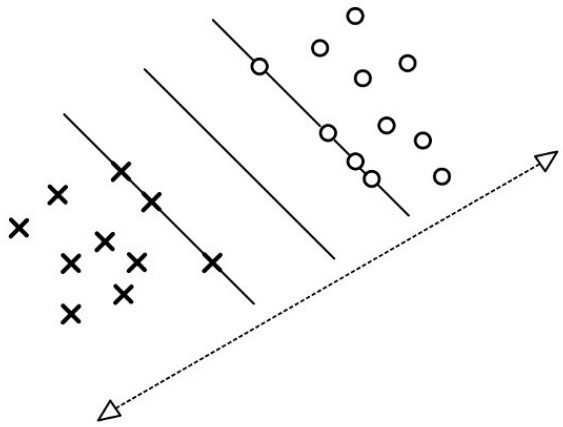

Figure 2.3: Linearly Separable and Soft Margin Plane Diagram

2.3 Linearly Separable SVM

Given a set of data, $T = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathbb{R}^n \times Y)^L$, where $x_i \in \mathbb{R}^n$ represents the input vectors, $y_i \in Y = \{1, -1\}$ represents the two categories of output samples, L represents the total number of samples.

step 1: Select a penalty coefficient $C > 0$ and construct a convex quadratic programming problem.

$$\min_a \quad \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j (x_i \cdot x_j) a_i \cdot a_j - \sum_{j=1}^{l} a_j$$

$$\text{s.t.} \sum_{i=1}^{l} y_i a_i = 0$$

$$0 \le a_i \le \zeta \quad \text{for} \quad i = 1, \ldots, l$$

The optimal solution obtained after solving is

$$a^* = (a_1^*, a_2^*, ..., a_l^*)^T$$

step 2: In the interval $(0, C)$, select the component a_j^* of a^* , and calculate to obtain b^* as:

$$b^* = y_j - \sum_{i=1}^{l} a_i^* y_i (x_i \cdot x_j)$$

step 3: Construct the differentiating hyperplane $(w^* \cdot x) + b^* = 0$, calculate the decision function as:

$$f(x) = \text{sgn}(g(x))$$

Where, $g(x) = \sum_{i=1}^{l} y_i a_i^* (x_i \cdot x) + b^*$

2.4 Approximate linear separability problem

In fact, in many practical problems, linear separability cannot be fully satisfied. Although some issues are separable due to various reasons, there may still be some inconspicuous fuzziness, overlaps, or even the presence of outliers [5]. This situation may produce a significant impact on generating an optimal classification hyperplane.

We can balance the relationship between generalization performance and empirical risk by introducing a slack variable ξ_i. However, it's worth noting that this must be based on the premise that slack variables allow for some sample points to be misclassified.

First, we relax the constraints on the classification hyperplane $w \cdot x + b = 0$, resulting in new constraint conditions as follows:

$$\text{s.t.} \quad y_i(< w \cdot x_i > +b) \geq 1 - \xi_i \quad \text{for} \quad i = 1, 2, ..., n$$

This is when we can continue to utilize linear classification to address the problem. If $\xi_i \geq 1$, they are classified incorrectly; On the contrary, if $0 < \xi_i < 1$, the sample points can be correctly classified. Next, we can introduce a new objective function by adding $C \sum_{i=1}^{n} \xi_i$ to the original objective function $\frac{1}{2}\|w\|^2$. Therefore, the new objective function is:

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

Here, $C \sum_{i=1}^{n} \xi_i$ can also be referred to as the penalty term.

C is the penalty factor mainly studied in this article, and its value plays a vital role in generalization ability. Therefore, we can change the C value to balance the accuracy and generalization ability of the model. If we aim to find the optimal solution to this quadratic programming problem, we can address the problem by finding the saddle point of the Lagrangian function. In this case, the Lagrangian function is given by:

$$L(w, b, a) = \frac{1}{2} < w, w > +C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} a_i[y_i(< w \cdot x_1 > +b) + \xi_i - 1] - \sum_{i=1}^{n} \beta_i \xi_i$$

Next, through the Karush-Kuhn-Tucker (KKT) theorem, we can obtain that the optimal solution satisfies the following four conditions:

$$\frac{\partial L}{\partial \xi_i} = C - a_i - \beta_i = 0$$

$$a_i[y_i(w \cdot x_i) + b - 1 + \xi_i] = 0$$

$$a_i, \beta_i, \xi_i \geq 0$$

$$\beta_1 \cdot \xi_i = 0$$

Substituting these four conditions into the new objective function, we can obtain:

$$\max \quad L(a) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i,j=1}^{n} a_i a_j y_i y_j < x_i, x_j >$$

$$\text{s.t.} \quad 0 \leq a_i \leq C, \ \sum_{i=1}^{n} y_i a_i = 0 \quad \text{for} \quad i = 1, 2, ..., n$$

Then, we can proceed with the same process as linear classification.

2.5 Non-linearly Separable Problem

In our daily lives, most of the problems we encounter are linearly inseparable. In such cases, linear mapping functions cannot be used. Currently, non-linearly separable problems in a low-dimensional space into linearly separable problems in a high-dimensional space poses a challenging task. Here, we introduce a new objective function as well:

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

$$\text{s.t.} \quad y_i(wx_i + b) - 1 + \xi_i \geq 0 \quad \text{for} \quad i = 1, 2, ..., n$$

$$\xi_i \geq 0 \quad \text{for} \quad i = 1, 2, ..., n$$

The next step is to transform it into an unconstrained problem using the Lagrange multiplier method:

$$L(w, b, \xi, a, \mu) = \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} - \sum_{i=1}^{n} \mu_i \xi_i - \sum_{i=1}^{n} a_i [y_i(wx_i + b) - 1 + \xi_i]$$

$$\text{s.t.} \quad C - \mu_i - a_i = 0 \quad \text{for} \quad i = 1, 2, ..., n$$

$$a_i \geq 0, \ \mu_i \geq 0 \quad \text{for} \quad i = 1, 2, ..., n$$

Then, by applying the KKT condition $\frac{\partial L(w,b,a,\xi,\mu)}{\partial w} = 0$ once again, we can obtain:

$$w = \sum_{i=1}^{n} a_i y_i x_i$$

Therefore, the Lagrangian function has the following formula:

$$a_i(y_i(wx_i + b) - 1 + \xi_i) = 0$$

$$\mu_i\xi_i = (c - a)\xi_i = 0$$

Similar to the linearly separable SVM, it is not valid for any $a_i = 0$. Hence, we can determine b using the support vectors corresponding to $a_i > 0$.

When dealing with non-linear classification problems in the input space, we can introduce non-linear transformations and convert the problem into a linearly separable problem in high-dimensional features to solve. Then, Within the feature space with a high dimensionality, we can use linear SVM to analyze and solve the problem. A notable benefit of utilizing a linear SVM is that, in its dual problem, the classification decision function and the objective function solely depend on the inner product of instances; no necessitating the explicit definition of non-linear transformations is required. Instead, one alternative to the inner product is to employ a kernel function. Kernel functions enable us to compute the inner product between two instances after they have undergone non-linear transformations. In particular, $K(x_i, x_j)$ represents a function, or more precisely, a positive definite kernel. This suggests the existence of a mapping $\Phi(x)$ from the input space to the feature space, which is valid for any two instances in the input space, x and z, it satisfies the formula:

$$K(x_i, x_j) =< \Phi(x_i) \cdot \Phi(x_j) >$$

When solving the dual problem using a linear SVM, we can replace the inner product with the kernel function $K(x_i, x_j)$ to address the issue of a non-linear SVM. It's worth noting that $K(x_i, x_j)$ must satisfy Mercer's theorem. Here, the objective function for the non-linear SVM with the maximum margin is:

$$\max_a \ W(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j=1}^{n} a_i a_j y_i y_j K(x_i, x_j)$$

At this point, the classification function is:

$$f(x) = \text{sgn}(< w \cdot \Phi(x) > \cdot x + b) = \text{sgn}\left(\sum_{i=1}^{n} \hat{a}_i y_i K(x_i, x) + \hat{b}\right)$$

The subsequent optimization problem is transformed into:

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

$$\text{s.t.} \quad y_i(< w \cdot \Phi(x_i) > +b) \geq 1 - \xi_i \quad \text{for} \quad i = 1, 2, ..., n$$

Its dual problem is:

$$\max \quad L(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j=1}^{n} a_i a_j y_i y_j K(x_i, x_j)$$

$$\text{s.t.} \quad 0 \leq a_i \leq C, \sum_{i=1}^{n} y_i a_i = 0 \quad \text{for} \quad i = 1, 2, ..., n$$

In this way, it reverts to the essence of a quadratic programming problem, and as a result, the optimal classification function changes to:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} \hat{a}_i y_i K(x_i, x_j) + \hat{b}\right)$$

2.6 Kernel Functions in SVM

Kernel methods, as an algorithm, must also satisfy Mercer's theorem, that is, if there is $g(x) \in L_2(\mathbb{R}^n)$, $K(x, y) \in L_2(\mathbb{R}^n \times \mathbb{R}^n)$, then for any $g(x) \neq 0$ and $\int g(x)^2 dx < \infty$, the existence of $\iint K(x, y)g(x)g(y)\,dx\,dy \geq 0$ is true. Its computation relies entirely on the dot product between data points. At this point, the dot product can be replaced with a kernel function, which is used to calculate the dot product in a higher-dimensional feature space [6]. In recent years, due to the increasing popularity of SVM, kernel methods also have garnered more attention. Kernel functions are a class of essential mathematical tools that can quickly transform traditional dot product-based algorithms into non-linear methods [7]. As a result, kernel functions play a crucial role in many fields. It is precisely because of the introduction of the kernel function that SVM has also been made practical, which effectively avoids the complex operations caused by vector inner products in high-dimensional spaces [8]. In general, kernel functions map the original data from a low-dimensional space to a higher-dimensional space [9].

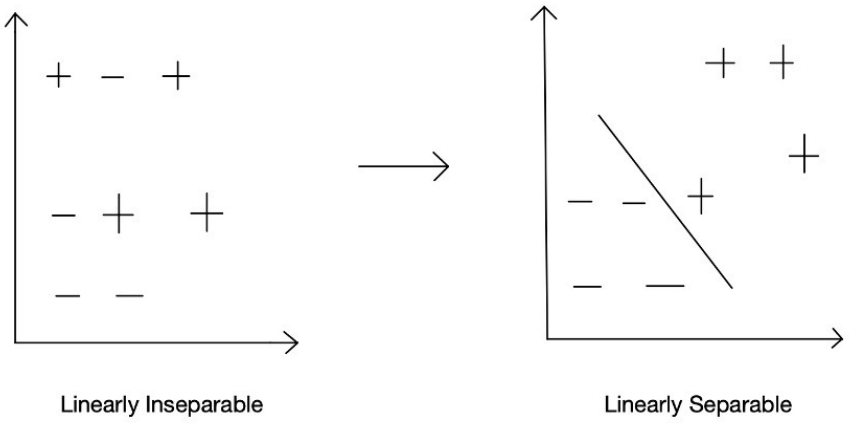

Figure 2.4: kernel function mapping visualization

2.6.1 Basic Theory of Kernel Functions

Let X be a subset or discrete set of Euclidean space $\mathbb{R}^n$, also called the input space. H represents the Hilbert space, serving as the feature space. If there is a mapping from X to H, denoted as $\phi(x) : X \to H$, and for any elements u and v within X, we have $K(u, v)$ satisfies the equation $K(u, v) = \langle \phi(u), \phi(v) \rangle$, then we can define $K(u, v)$ as a kernel function and $\phi(u)$ as the mapping function. Here, $\langle \phi(u), \phi(v) \rangle$ signifies the inner product of $\phi(u)$ and $\phi(v)$.

$K(u, v) = \phi(u) \cdot \phi(v)$ can be used to solve non-linear SVM classification problems. First, the inner product $\psi(u_i) \cdot \psi(v_j)$ is constructed in the original input space, and then a nonlinear transformation is performed on it. This algorithm no longer requires most of the computations to be transferred to a high-dimensional feature space, but can directly perform operations in the original input space, which is its key advantage. As shown in Figure 2.5, this figure shows the image of two-dimensional space mapped to three-dimensional space, making it easier to distinguish, which is also why kernel functions are often used for classification and clustering.

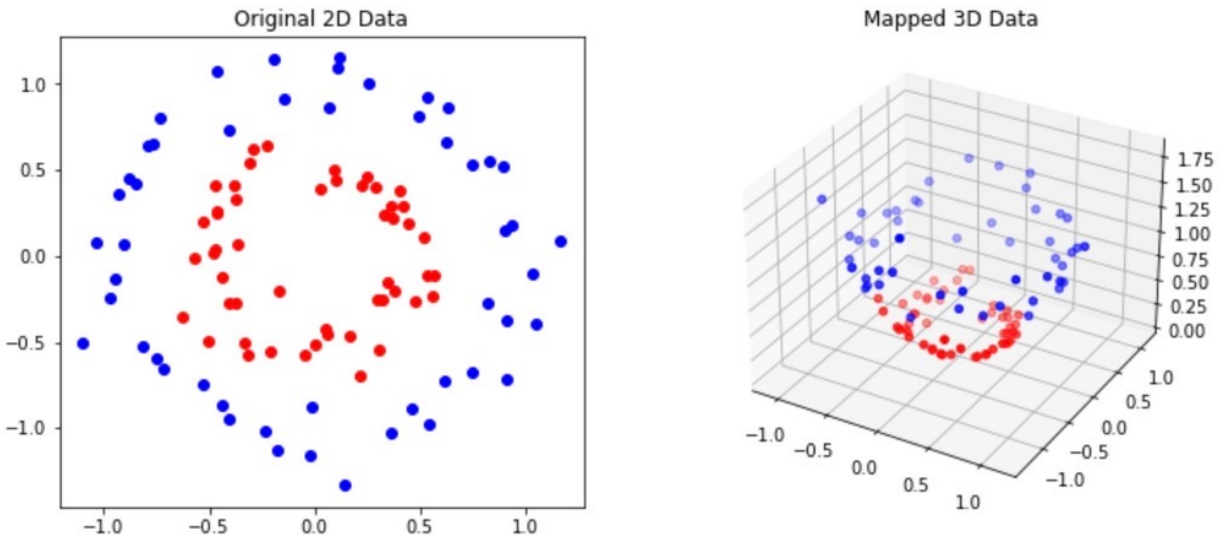

Figure 2.5: Example of Mapping Two-Dimensional Space to Three-Dimensional Space

Here, we can illustrate this concept with an example, given that mapping from a two-dimensional feature space to a three-dimensional space:

$$\Phi : u = (u_1, u_2) \rightarrow \Phi(u) = (u_1^2, u_2^2, \sqrt{2}u_1u_2) \in F = \mathbb{R}^3$$

And the inner product in the feature space is:

$$< \Phi(u), \Phi(v) >=< (u_1^2, u_2^2, \sqrt{2}u_1u_2), (v_1^2, v_2^2, \sqrt{2}v_1v_2) >=< u, v >^2$$

Then we can obtain the kernel function:

$$K(u, v) =< u, v >^2= \Phi(u)^T\Phi(v)$$

However, it's worth noting that the kernel function can only compute the inner product of the mapping. Therefore, even if we switch to a 4-dimensional feature space: $\Phi(u) = (u_1^2, u_2^2, u_1u_2, u_2u_1) \in F = \mathbb{R}^4$ we can still get the kernel function just calculated. It can be seen that the choice of kernel function in the feature space mapping of SVM is not unique.

The fundamental role of kernel functions in SVM is to transform training samples from the original problem space into linearly separable training samples in the feature space [10].

Therefore, the key to the SVM algorithm lies in the choice of the kernel function. To date, four primary forms of kernel inner product functions have been researched: Polynomial kernel function, Linear kernel function, Gaussian kernel function, and Sigmoid kernel function. By using different kernel functions for different types of data points, various attribute mappings from linear to non-linear have been achieved.

2.6.2 Polynomial Kernel Function

The definition of the polynomial kernel function is given by:

$$K(x, x_i) = (1 + x \cdot x_i^T)^d$$

where d represents the degree.

Polynomial functions exhibit directional characteristics, meaning that the output results are influenced by the orientation of two vectors in the low-dimensional space due to the dot product in the kernel. Nonetheless, given the abundance of parameters to consider for the polynomial kernel function, as the polynomial order becomes high, the elements in the kernel matrix tend to approach either infinity or infinitesimal values. This behavior can lead to numerical instability, significantly complicating the computational process.

2.6.3 Linear Kernel Function

Without adding non-linear mapping, the linear kernel function can be employed for the purpose of mapping data from the original feature space into a feature space with a higher dimension. Its definition is:

$$K(x, x_i) = x \cdot x^T$$

The dimensions of the feature space and the input space of the linear kernel function are the same, as are the feature values of every vector. In particular, when the data points in the feature space are linearly separable, the linear kernel function is frequently appropriate for processing objects represented by a large number of fixed-length features because it

executes an inner product operation in the original feature space. At the same time, the linear kernel function can guarantee the formal unification of the "problem before mapping" and the "problem after mapping," allowing common expressions to be used initially and subsequently swapped into various kernels when defining mathematical formulas or writing code. This dramatically reduces the complexity of code writing and algorithm development.

2.6.4 Gaussian Kernel Function

The Gaussian kernel function is a highly versatile kernel function that can be applied to various types of sample distributions by selecting appropriate parameters. Additionally, it is also frequently utilized in SVM to solve nonlinear mapping issues [11]. Its definition is:

$$K(x_i, x_j) = \exp(-\frac{x_i - x_j^2}{2\sigma^2})$$

Where, σ represents the width of the kernel function. This formula can be changed to produce a different variant of the Gaussian kernel function by replacing the parameter $\gamma = \frac{1}{2\sigma^2}$:

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

In order to optimize the objective function, the relaxation factor $\xi(i)$ is introduced in the SVM algorithm. Therefore, there is a constraint relationship between the objective function and the relaxation factor, which can usually be expressed as:

$$\min \quad \frac{1}{2}w^2 + C\xi(i)$$

$$y_i[wx_i + b] > 1 - \xi(i) \quad \text{for} \quad i = 1, 2, ..., n$$

2.6.5 Sigmoid Kernel Function

The sigmoid kernel function is defined as follows:

$$K(x_i, x_j) = \tanh(v(x_i \cdot x_j) + c)$$

where v represents a scalar quantity and c represents offset.

In the sigmoid kernel function, v and c are adjustable parameters, and the values of v and c can be adjusted according to specific problems to obtain the best model. It's important to note that the sigmoid kernel function may not be suitable for all non-linear separable problems. And overly large or small input values have the potential to approach saturation and result in the non-existent gradient issue.

2.7 The Choice of Kernel Function

Different types of kernel functions have their own advantages and limitations, and these factors also lead to their non-linear modeling capabilities. There are numerous varieties of kernel functions, so explaining their respective properties in detail can be complicated. However, kernel functions generally can be summarized into two main types: global kernel functions and local kernel functions [12]. Data types will be the main factor in how to choose them. If the data is evenly distributed in the entire input space, it is more necessary to use the global kernel function. If the data exhibits clear local structures, then local kernel functions should be chosen.

2.7.1 Global Kernel Functions

A kernel function that can be represented as a function of $K(x_i, x_j) = f(< x_i, x_j >)$ is called a rotation-invariant kernel, where $f : D \to \mathbb{R}$ is a unary real-valued function ($D \in \mathbb{R}$). This is a type of global kernel function. To be a Mercer kernel function, it must satisfy that for any $\varepsilon \geq 0$, the following formula holds:

$$K(\varepsilon) \geq 0$$

$$K'(\varepsilon) \geq 0$$

$$K(\varepsilon) + \varepsilon K'(\varepsilon) \geq 0$$

Therefore, we can conclude that the polynomial kernel function $K(x, x_i) = (1 + x \cdot x_i^T)^d$ is a representative global kernel function.

No matter where it is in the input space, the polynomial kernel function can have similar responses, as shown in Figure 2.6, which fully demonstrates the global properties of a simple polynomial kernel function.

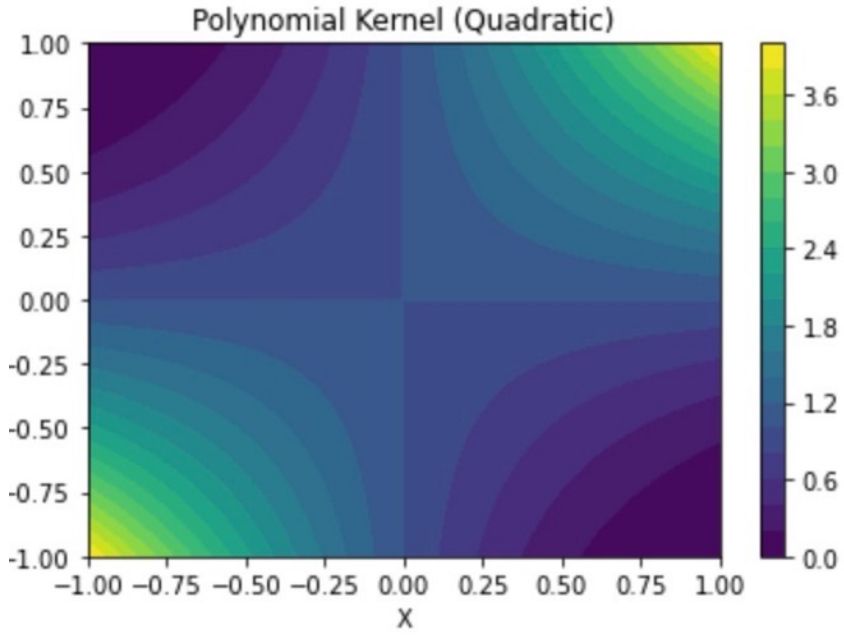

Figure 2.6: The response image of a quadratic polynomial kernel function

However, different polynomial kernel functions may result in various shapes or features since their shape and properties change as their degrees and coefficients change.

2.7.2 Local Kernel Functions

A translation-invariant kernel is a type of local kernel function, and we refer to a kernel function that can be represented as a function of $K(x_i, x_j) = f(x_i - x_j)$ as a translation-invariant kernel. Its determination theorem states: Let $f : X \to \mathbb{R}$ be a continuous function and bounded integrable. Then the necessary and sufficient conditions for judging that

$K(x_i, x_j) = f(x_i - x_j)$ is a kernel function is $f(0) > 0$, which satisfies the Fourier transform

$$F(w) = \int_{-\infty}^{+\infty} f(x) e^{-i(w \cdot x)} dx > 0$$

Therefore, we can see the Gaussian kernel function, which represented as $K(x_i, x_j) = \exp(-\gamma \left| x_i - x_j \right|^2)$, is a typical example of a local kernel function.

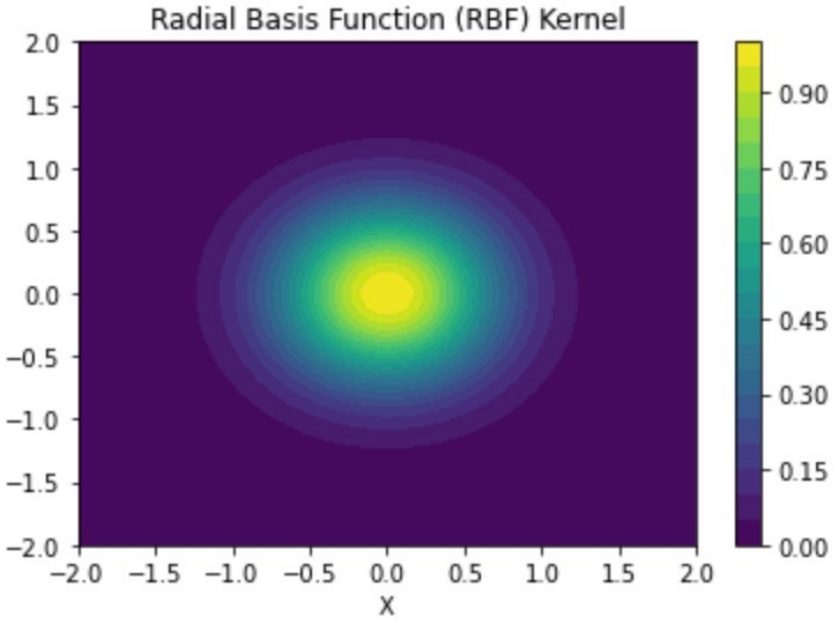

Figure 2.7: The response image of the Gaussian kernel function

This image showcases the response of the Gaussian kernel function in a two-dimensional input space and also shows the local properties of the Gaussian kernel function. It takes the center point as the center, generates a high-response local structure nearby, and then gradually reduces the response away from the center point. This characteristic allows local kernel functions to capture the local structure of data effectively.

2.7.3 A Comparison of The Two Types of Kernel Functions.

Here, we also use Gaussian and polynomial kernel functions for graphing as two common local and global kernel functions.

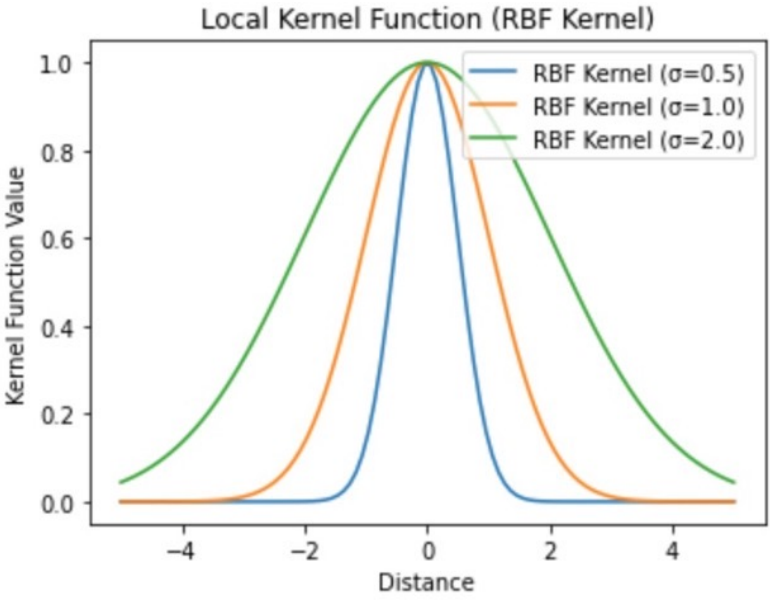

Figure 2.8: The Gaussian kernel function curve

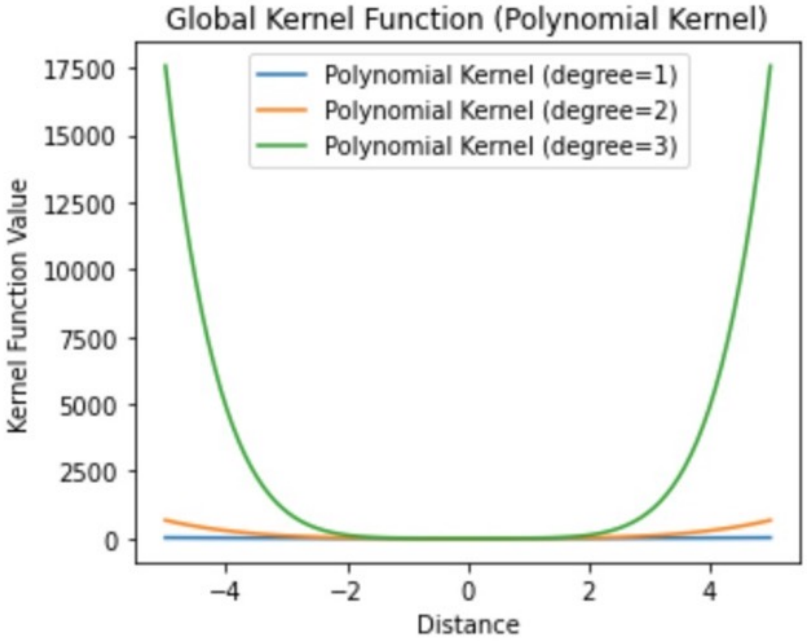

Figure 2.9: The polynomial kernel function curve

Figure 8 illustrates the images under different Gaussian kernel function parameters (σ) values. Each curve represents a specific σ value. The smaller the σ value, the steeper the curve, and the center point has a more significant impact on the weight of the data points. The Gaussian kernel function has a peak near the data point and then decreases rapidly, so it has a greater impact on the data point in a local range. Larger σ values lead to smoother kernel function curves, while smaller σ values result in steeper curves.

Figure 9 presents the images under different polynomial kernel function parameters (polynomial degrees). Each curve represents a specific polynomial degree, and different polynomial degrees result in varying kernel function shapes. Polynomial kernel functions of higher degrees still have higher values far away from the center point, while polynomial kernel functions of lower degrees decrease rapidly away from the center point. The high degree of the polynomial kernel function allows the kernel function to have a greater weight on the data points farther away from the center point.

These two figures emphasize the impact of different kernel function parameters on the shape of the kernel function and the distribution of data point weights to help understand the different properties and applications of the kernel function. In general, choosing the appropriate kernel function and parameters is very important to solve a specific problem and model the data.

2.8 SVM Parameter Selection

Essentially, finding the solution (a and b) that maximizes the following equation is the first step in training an SVM. Moreover, the constants C and kernel function parameters are chosen to optimize the model's generalization capability estimates [13].

$$\max_{a} \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i,j=1}^{l} a_i a_j y_i y_j K(x_i, x_j) \quad \text{for} \quad 0 \leq a_i \leq C$$

Therefore, we can view parameter selection as a minimization or maximization problem and then perform the following steps:
(1) Initial setting of the error penalty parameter C and the kernel function's intrinsic parameters.
(2) Maximize this expression to obtain the values of a and b.
(3) Minimize the estimate of generalization ability by updating the C value and kernel function's intrinsic parameter values.
(4) If a suitable estimate value is obtained, the computation ends. Otherwise, continue to repeat step (2).

For an SVM, If the error penalty parameter C is increased indefinitely, it will seem that the C value change does not affect the classification performance when the SVM has no boundary support vectors. At this time, the classification performance of SVM may be affected by other factors, such as the selection of kernel function.

To find a function that satisfies Empirical Risk Minimization (ERM), it must determine the VC dimension of the function. Put otherwise, once the ERM is determined, the decision function of SVM can provide the smallest confidence interval. However, because the data distribution of different subspaces is also different, the optimal SVM classifier that satisfies the structural risk minimization (SRM) principle shows different classification effects for different feature subspaces. In the SVM model, optimizing the penalty error parameter C and the parameters σ in the kernel function are crucial to the model's performance. By optimizing these parameters, we can obtain the global optimal solution, which is of great significance to the classifier's learning ability, classification accuracy, and generalization performance. The SRM principle states that various combinations of C and kernel parameters can yield the SVM model's risk upper bound, allowing for the creation of the best SVM classifier, the successful avoidance of over-fitting or under-fitting issues, and an improvement in generalization capacity.

At the same time, the penalty factor C is used as a trade-off factor between the com-

plexity of the balancing algorithm and sample misclassification. When C is massive, the model does not allow for misclassification; that is, the samples must be linearly separable, which can lead to a significant increase in model complexity. On the contrary, when C is very small, the SVM model's penalty for misclassification is almost negligible. In this case, the model is only concerned with maximizing the interval without caring about classification correctness, which can lead to an increase in the empirical value at risk of the model and, thus, to non-convergence. These two situations can lead to overfitting and underfitting problems, respectively. In addition, the complexity of the model reaches an upper limit when the value of C increases to a certain level, at which time the empirical risk and generalization ability hardly change significantly. Therefore, a reasonable choice of C is crucial for obtaining an ideal SVM classifier.

Furthermore, the performance of SVM is also affected by the kernel function and kernel parameters. With no clear theoretical guidance, choosing the kernel function and kernel parameters for SVM classification algorithms has to be a challenging task. To identify which kernel function to use, numerous researchers have carried out a vast number of experiments. The practice has demonstrated that RBF usually has a good classification effect and generalization ability compared to other kernel functions. So, the parameter settings in the RBF kernel function have been one of the focal points of research. To get the best classification results, it is typically necessary to test various values of the RBF kernel function parameters in various trials. As a result, choosing an appropriate kernel function and adjusting the appropriate penalty factor C as well as the kernel parameters are of great significance for optimizing the classification model.

2.9 SVM Parameter Optimization Methods

The goal of SVM algorithms is to determine the linear or non-linear decision boundaries that achieve optimal classification in n-dimensional space. Two of the key parameters are

C and Kernel, which usually have default values, but they often lead to low classification accuracy [14]. To solve the parameter optimization problem, we can rely on "Tree-Structured Parzen Estimator Optimization for Support Vector Machines (TPEOSM)."

In extreme value issues, Bayesian Optimization (BO) can be used for functions whose expressions are uncertain. The two primary parts of this black-box optimization approach are a collection function and a Gaussian process regression. One benefit of using Bayesian optimization is that it only requires a limited number of samples to infer the maximum value of the function, and typically only requires a relatively small number of sampling points.

Gaussian process regression is used to estimate the probability distribution of a function value at any point based on the function value at a series of sample points. It first selects sample points evenly within the area and ensures that they satisfy the multidimensional normal distribution. These points are then initialized and finally used as candidate solutions. As in the following equation:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \sim N \left(0, K \begin{bmatrix} (x_1, x_1) & \cdots & (x_1, x_n) \\ \vdots & & \vdots \\ (x_n, x_1) & \cdots & (x_n, x_n) \end{bmatrix} \right)$$

Here, K represents the covariance matrix. Then according to the posterior formula, we obtain the updated value y_*:

$$P(y_*|y \sim N(K_* K^{-1} y, K_{**} - K^{-1} K_*^T))$$

Here, K_*represents the covariance matrix of the original samples, and K_{**} is the covariance matrix of the newly added samples.

Finally, the updated candidate solution is given by:

$$\begin{bmatrix} y \\ y_* \end{bmatrix} \sim N \left(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix} \right)$$

Based on the results of the Gaussian process regression, we can construct the acquisition function. Subsequently, once the extremes of the acquisition function have been identified,

Bayesian optimization will next decide the subsequent sampling points. And then these sampling points will also eventually be returned as the estimated optimal values of the function.

$$x_{n+1} = \arg\max EI_n(x)$$

$$EI_n(x) = E_n[[f(x) - f_n^*]^+] = (\mu - f_n^*)\left(1 - \phi\left(\frac{f_n^* - \mu}{\sigma}\right)\right) + \sigma\psi\left(\frac{f_n^* - \mu}{\sigma}\right)$$

One could think of the TPE as an enhanced and more effective variant of Bayesian optimization. It improves the original algorithm by introducing a tree structure to construct the solution space and simultaneously uses the Parzen estimator to calculate the probability density.

Given that $p(x|y)$ is the conditional probability with hyperparameter x and model loss y, and y^* is the threshold. Let $l(x)$ and $g(x)$ represent the probability density functions for learning above the threshold and below the threshold, respectively [15]. Then we can derive the formula:

$$p(x|y) = \begin{cases} l(x) & if\ y < y^* \\ g(x) & if\ y \geq y^* \end{cases}$$

To make the optimization process for "Expected Improvement" (EI) in the TPE algorithm simpler, the parameter $p(x, y)$ is chosen as $p(y)\ p(x|y)$. The Bayesian formula can be used to translate it since $p(y|x)$ is not available:

$$EI_{y_*}(x) = \int_{-\infty}^{y^*}(y^* - y)p(y|x)dy = \int_{-\infty}^{y^*}(y^* - y)\frac{p(x|y)p(y)}{p(x)}dy$$

Next, let $\gamma = p(y < y^*)$ to partition $l(x)$ and $g(x)$.

Therefore, EI can be simplified to:

$$EI_{y_*}(x) = \frac{\gamma y^* - l(x)\int_{-\infty}^{y^*}p(y)dy}{\gamma l(x) + (1-\gamma)g(x)} \propto \left(\gamma + \frac{g(x)}{l(x)}(1-\gamma)\right)^{-1}$$

It is evident that in order to get the highest possible EI, we must make sure that $g(x) \setminus l(x)$ is reduced to finish the necessary amount of iterations. Additionally, we can obtain the

candidate x^* for maximizing EI at each iteration:

$$x^* = \arg\max \ EI_{y^*}(x)$$

At this point, we can obtain the optimal value and optimal parameters of the objective function.

CHAPTER 3

Experiment

This research paper uses the Iris dataset from the UC Irvine Machine Learning Repository [16].

This dataset contains 150 sample points with four eigenvalues each. In this study, only two features, sepal length and sepal width, are used to represent the feature classification of the iris under different kernel functions.

First, we classify the complete 4-dimensional feature dataset (the original iris dataset) to evaluate the accuracy of different kernel functions on the prediction and test sets. Then, the TPE method is used to optimize the feature values. This process will help us select the combination of feature parameters applicable to the Iris dataset more efficiently to achieve better classification accuracy. Finally, feature classification for the iris dataset with different parameter settings for the same kernel function is performed and cross-validated.

3.1 Classification of Iris features with different kernel functions

Two characteristics, the length and width of the sepal, served as the foundation for our classification in order to improve the clarity and understandability of the classification results. This classification is called "feature classification," which reduces the feature information of sample points to the two most critical and representative dimensions. Subsequently, different kernel functions are selected for classification, and the classification effect is visualized as a two-dimensional graph. Here, we set the initial value of the error penalty parameter C to 1.

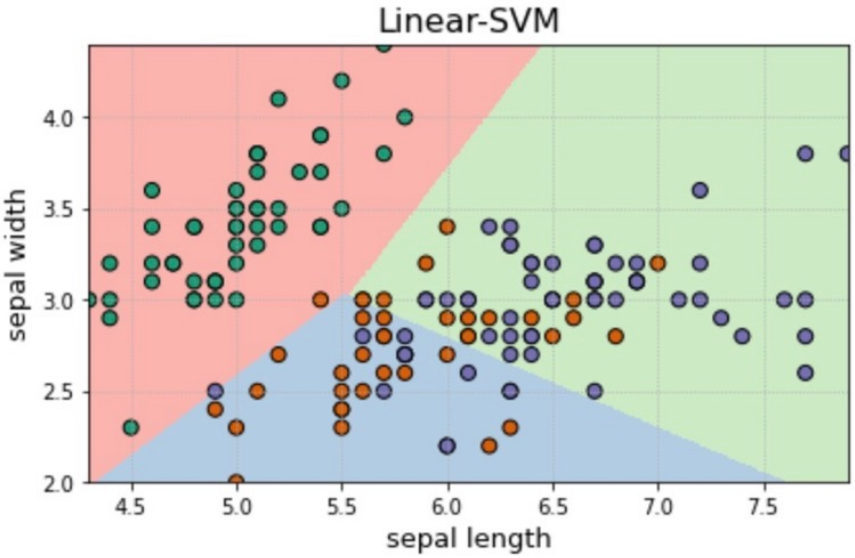

Figure 3.1: Linear kernel function feature classification

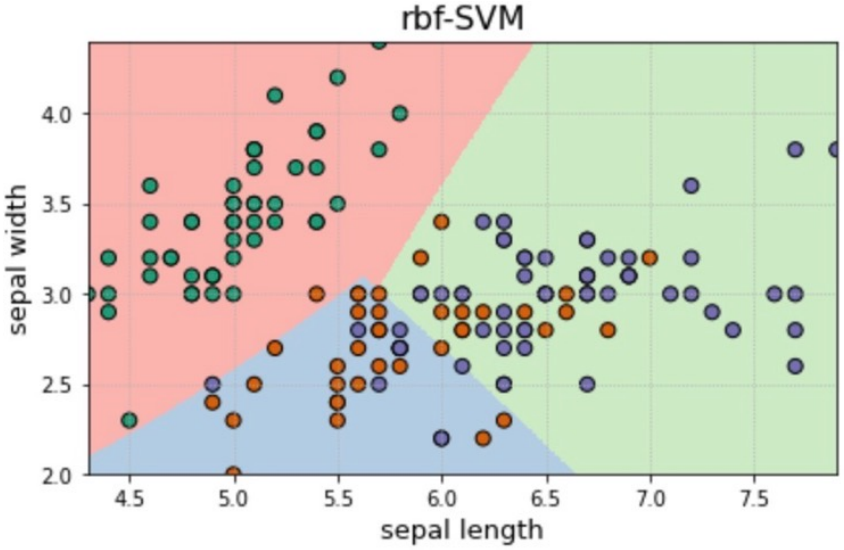

Figure 3.2: Gaussian kernel function feature classification

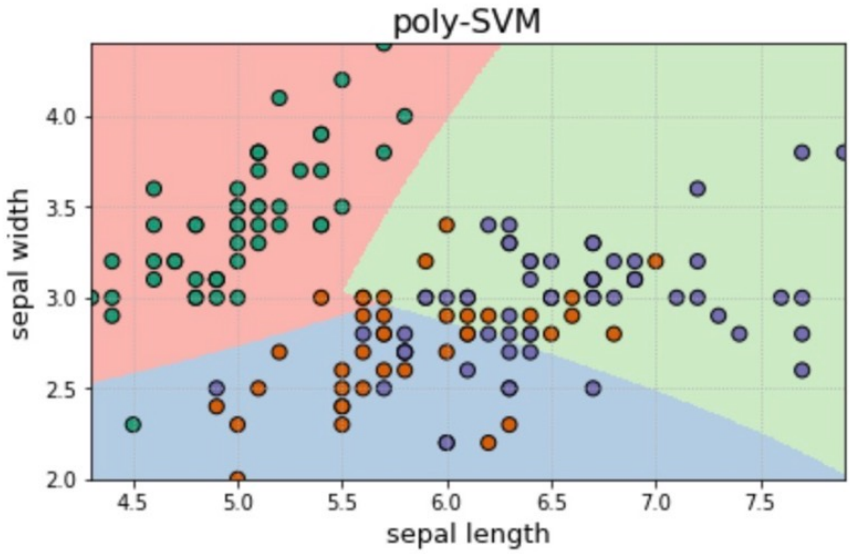

Figure 3.3: Polynomial kernel function feature classification

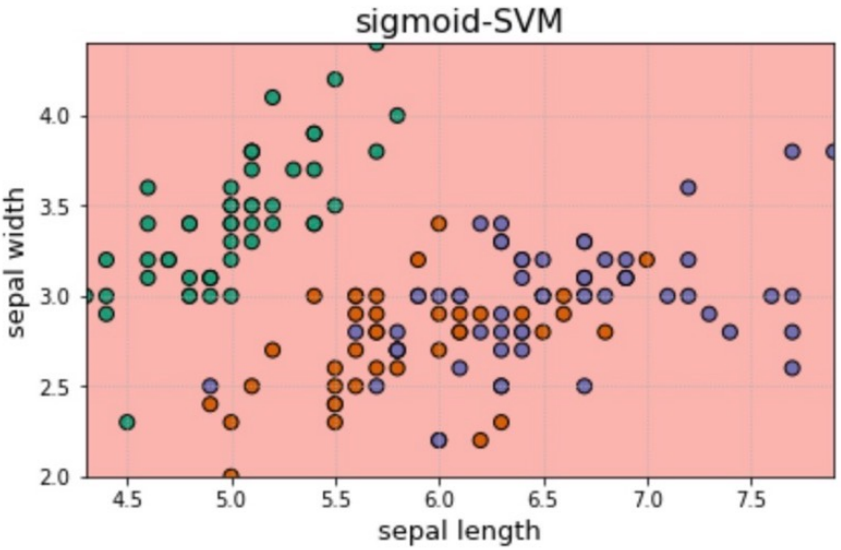

Figure 3.4: Sigmoid kernel function feature classification

The image demonstrates how efficiently the Gaussian kernel function operates to produce superior classification results. This further supports the earlier claim that the Gaussian kernel function is typically sufficient to get superior outcomes in feature classification issues. This result gives us strong guidance on selecting the right kernel function to enhance the classification model's functionality and capacity for generalization.

3.2 Comparison of Different Kernel Function Models

We divide the dataset proportionally into training and test sets and then create SVM classifiers with four different kernel functions, namely linear, Gaussian, polynomial, and sigmoid kernels. These models are used in the subsequent part to classify the dataset. To train the SVM model using each of the four different kernel functions, choose a value for parameter C, and each model's training duration is noted so that the models' performances may be compared.

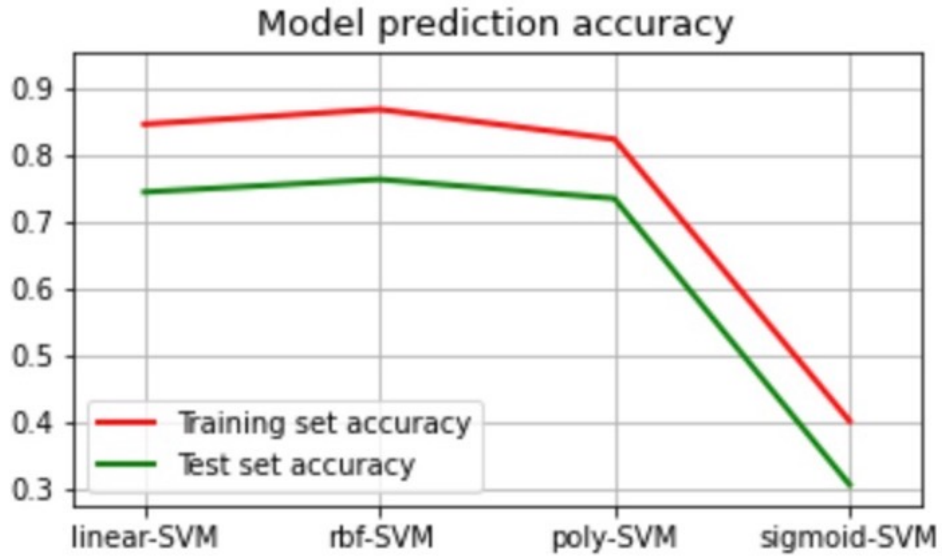

Figure 3.5: Model Prediction Accuracy

Figure 3.6: Model Training Time

From the images, we can observe that under the same parameters, the Gaussian kernel function achieves the highest testing accuracy and has an advantage in model training time.

3.3 Test Set Accuracy with Different Parameters

We chose four different kernel functions and set different values of the parameter C to compare their accuracy on the test set.

C	Linear	rbf	poly	sigmoid
1	75.1%	78.5%	72.8%	33.9%
2	73.8%	79.6%	73.5%	32.1%
3	73.3%	75.3%	71.3%	32.3%
4	73.5%	73.85%	71.1%	31.3%

Table 3.1: Test Set Accuracy for Different Kernel Functions with Different Parameters

From Table 3.1, we can observe that the Gaussian kernel function consistently achieves

higher accuracy on the test set across various parameter settings than the other three kernel functions. Therefore, the Gaussian kernel function outperforms the other kernel functions in this case.

3.4 Using the TPE Method to Optimize Parameters

We determine the optimal hyperparameter combination for the SVM model based on the Iris dataset using the TPE approach for hyperparameter search here. And then train the model and assess its performance.

Step 1: define an objective function for hyperparameter optimization to maximize the classification accuracy of the model on the training set. Optuna will use the Bayesian optimization algorithm to continuously improve the performance of the objective function so that we can create an Optuna study object that searches for the best hyperparameter combination of the SVM model in a certain number of experiments. Then, output the values of the optimal hyperparameters. Here, our output value is:

$$\text{kernel} : \text{rbf},\ C : 1.9863322570254014,\ \gamma : 3.7351724209466353$$

```
# print the best hyperparameters.
best_params = study.best_params
print(best_params)
```

```
{'kernel': 'rbf', 'C': 1.9863322570254014, 'gamma': 3.7351724209466353}
```

Figure 3.7: Print the best hyperparameters

Step 2: draw the optimization history to show the changes in the objective function value in each trial.

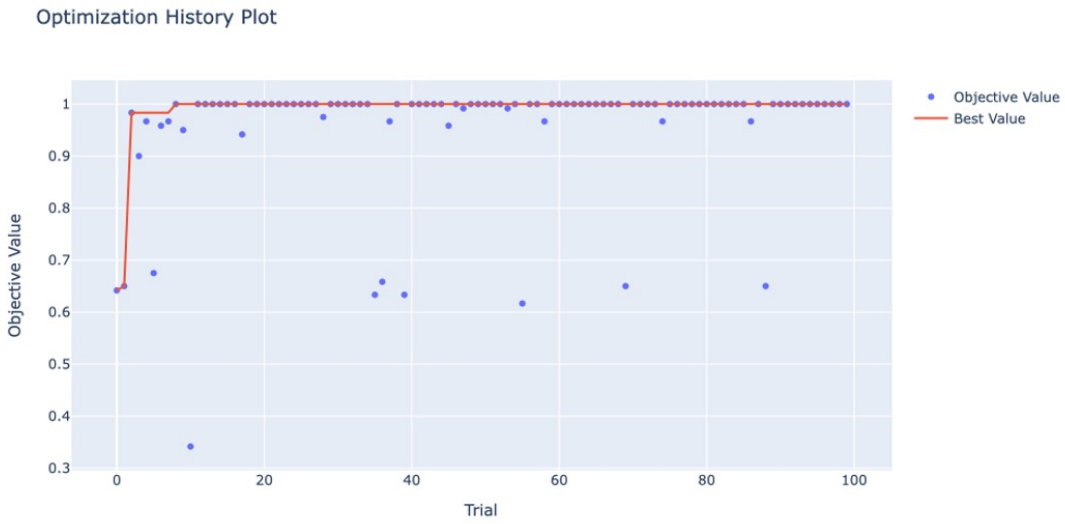

Figure 3.8: Optimization history plot

The blue points in the figure 3.8 represent the objective function value, and the red line represents the optimal value. The image shows that the curve stabilizes after about ten trials, indicating that the hyperparameter search may have found a good hyperparameter combination.

Step 3: train the model using optimal hyperparameters. For the given SVC model, we get the hyperparameter combination value:

$$C = 1.9863322570254014,\ \gamma = 3.7351724209466353$$

```
# Train the model using the best hyperparameters.
clf = SVC(**best_params)
clf.fit(X_train, y_train)
```

```
SVC(C=1.9863322570254014, gamma=3.7351724209466353)
```

Figure 3.9: Train the model using the best hyperparameters

Step 4: evaluate the model's performance on the training and test sets, obtaining an accuracy of 1.0 on the training set and 0.9666666666666667 on the test set.

```
# Print the classification accuracy on the training and test sets.
train_accuracy = accuracy_score(y_train, clf.predict(X_train))
test_accuracy = accuracy_score(y_test, clf.predict(X_test))
print(f"Training accuracy: {train_accuracy}")
print(f"Testing accuracy: {test_accuracy}")
```

```
Training accuracy: 1.0
Testing accuracy: 0.9666666666666667
```

Figure 3.10: Print the classification accuracy on the training and test sets

It can be seen from this plot that the model performs well in the iris data set and can effectively classify sepal length and width.

Step 5: draw the training set and test set fitting curves to show the comparison between the model's prediction results on the training set and the test set and the true values.

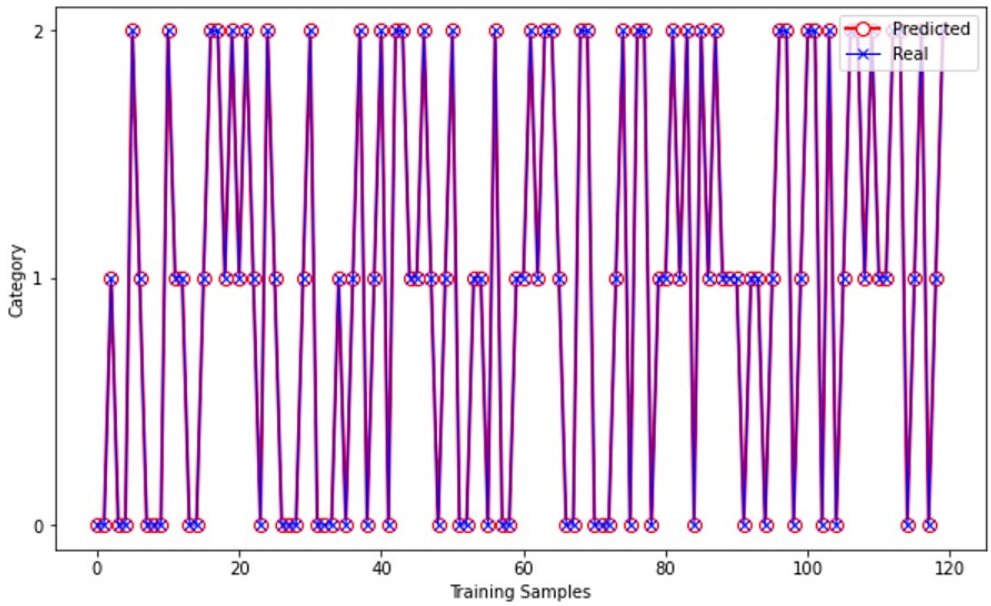

Figure 3.11: Training set fitting curve

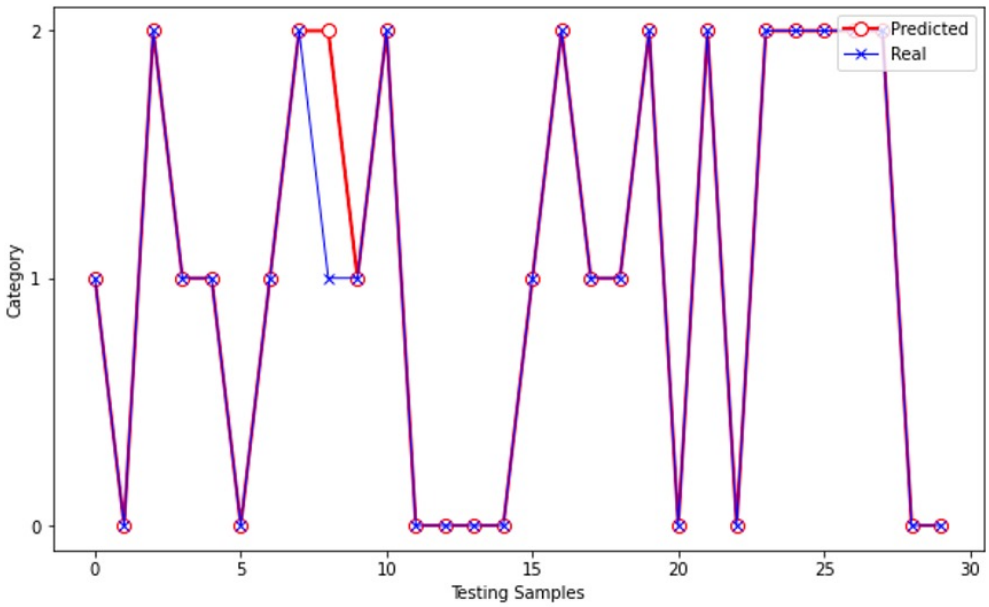

Figure 3.12: Testing set fitting curve

As you can see from the image, the model performs very well on the training data, and the training set curve fits perfectly. However, in the test set data, the curve fit is slightly biased. This indicates that there may be an overfitting problem. If we want to improve model performance and generalization ability, we can take measures such as simplifying the model and introducing regularization to reduce overfitting.

3.5 Impact of Different Parameter Values on Feature Classification

According to the above experimental comparison, we can observe that the Gaussian kernel function has a better classification effect. So here, we first select the kernel function as the Gaussian kernel function, then set different C values, C=(0.1, 1, 10, 100), and finally compare its impact on feature classification by observing the image.

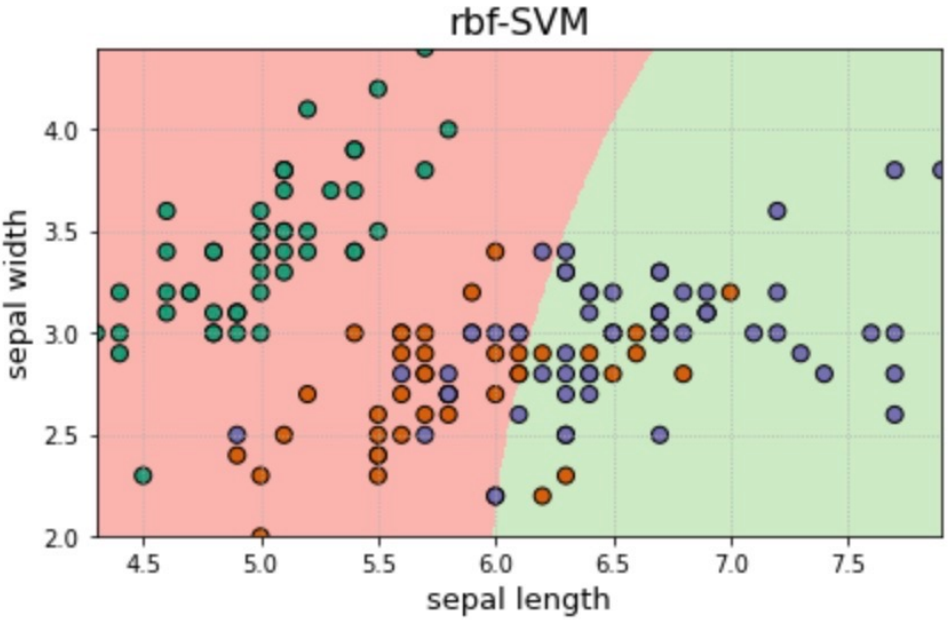

Figure 3.13: C=0.1

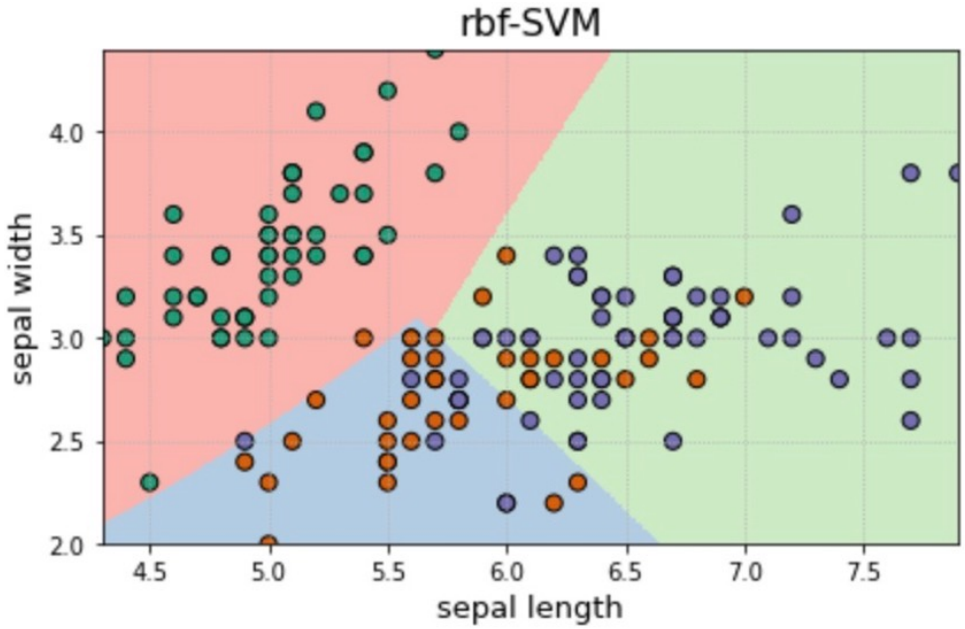

Figure 3.14: C=1

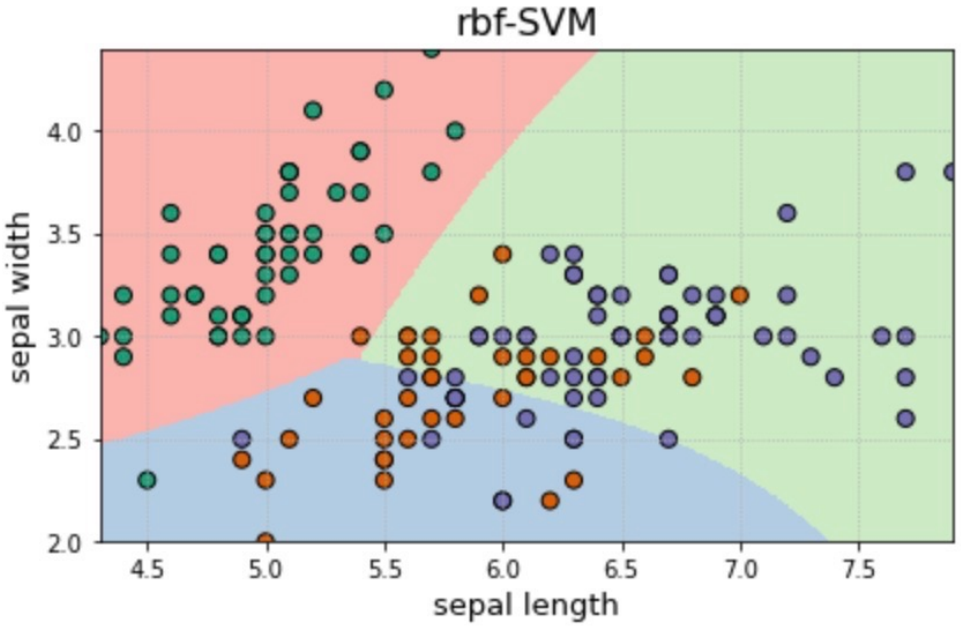

Figure 3.15: C=10

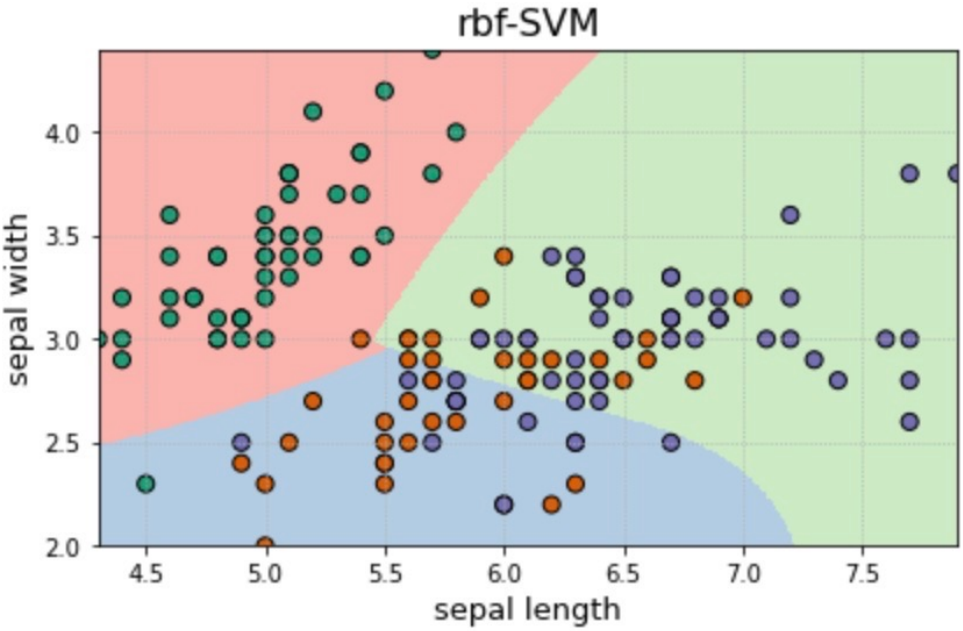

Figure 3.16: C=100

According to the image, we can find that the smaller the C value, the better the generalization ability. However, due to the ample error tolerance space of the model, underfitting problems are prone to occur. The larger the C value is, the model is too complex and cannot adapt to unseen data, resulting in poor generalization ability. When C exceeds a certain value, the generalization ability no longer changes significantly. Since the gamma parameter value will also affect the model's performance, we can find a suitable gamma value at this time and form a hyperparameter combination with the C value. A smaller gamma value will result in a smoother decision boundary, and a larger one will result in a more complex decision boundary. Therefore, the selection of C and gamma values must be within a reasonable range. Whether it is too large or too small, it may lead to a decrease in generalization performance. To find the best C and gamma values, we can use the cross-validation.

3.6 Cross-validation

By using cross-validation, we can assess how well the model performs under various C and gamma value combinations to determine which hyperparameter configuration best fits the data and yields the greatest prediction performance.

Here, we use nested loops to iterate over different combinations of C and gamma. For each combination, create an SVM classifier, use the RBF kernel function, and set the corresponding C and gamma values. Determine the C value (0.1, 1, 100) and the gamma value (0.1, 1, 100). Then, train the model on the training set.

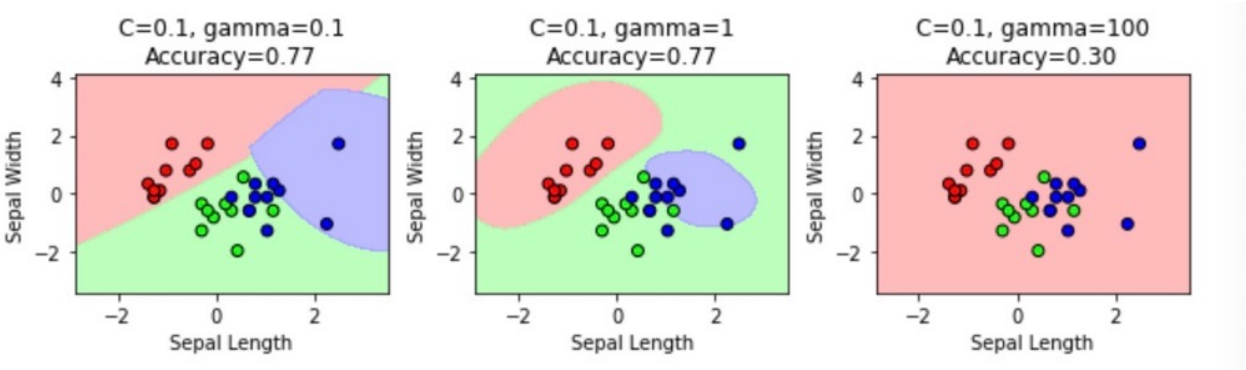

Figure 3.17: C=0.1, gamma=(0.1,1,100)

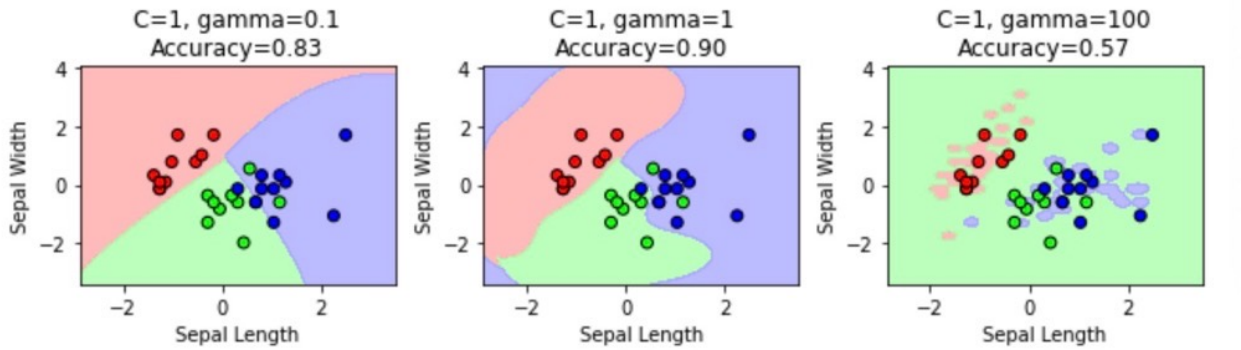

Figure 3.18: C=1, gamma=(0.1,1,100)

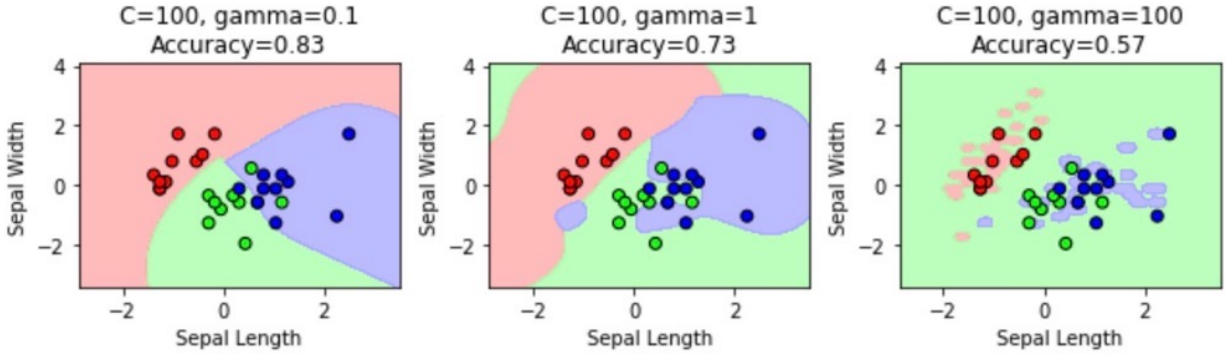

Figure 3.19: C=100, gamma=(0.1,1,100)

By observing the images, we can see the differences in the models under different combinations of C and gamma values:

1. When C=0.1, whether the gamma is 0.1, 1, or 100, the model's performance is not very good, and the accuracy is relatively low. This indicates that the C value is too small, and the model has greater fault tolerance, resulting in underfitting of the model.

2. When C=1, the model's performance significantly improves, and higher accuracy is obtained regardless of whether gamma is 0.1, 1, or 100. This shows that a moderate C value can balance the model's fault tolerance and overfitting problems, improving performance.

3. When C=100, the model's performance sometimes decreases slightly. For example, when gamma=0.1, the accuracy remains at 0.83, but when gamma=1 and 100, the accuracy drops to 0.73 and 0.57. This suggests that larger C values may sometimes cause the model to overfit.

Next, use GridSearchCV for grid search and cross-validation. It tries all possible hyperparameter combinations and uses 5-fold cross-validation to evaluate the performance of each combination.

```
Best parameters found:  {'C': 1, 'gamma': 1, 'kernel': 'rbf'}
Best cross-validated score: 0.96
Test set accuracy: 1.00
```

Figure 3.20: Cross-validation results

The best hyperparameter configuration, as determined by the cross-validation results, is as follows: $C = 1, \gamma = 1$, and kernel function type is RBF. The average cross-validation score under this optimal configuration is 0.96, which shows that the model functions well on the training set. The model's accuracy on the test set is 1.00, indicating that it also performs well on data that has not been seen before and can generalize to new data without experiencing issues with over- or under-fitting.

CHAPTER 4

Conclusion

4.1 Conclusion

This study first introduces the background and importance of SVM. In the era of big data, choosing the appropriate kernel function and correctly setting the penalty factor and kernel parameters are crucial for optimizing classification models. Selecting a proper kernel function can not only improve the running speed of the model but is also of great significance for the efficient classification of data. This paper selects two features in the Iris data set for experiments. It compares the performance of the Gaussian kernel function with other kernel functions through many experiments. The results show that the Gaussian kernel function performs well, with the shortest training time and the highest testing accuracy. Therefore, it is evident that the Gaussian kernel function can effectively approximate the expansion of infinite dimensions and is suitable for processing complex data.

In addition, this study analyzed the impact of parameters on SVM performance. Experimental results show that the value of parameter C has a direct effect on feature classification. A more considerable C value will lead to a decrease in generalization ability. It may cause overfitting problems, while a smaller C value will help improve generalization ability but may lead to underfitting. Therefore, choosing the appropriate parameter configuration is crucial to the performance of SVM. At the same time, the TPE method using Bayesian optimization tools was introduced for hyperparameter optimization, and the optimal hyperparameter combination was finally selected and cross-validated, further improving model performance.

4.2 Future Enhancement

As the core content of machine learning, SVM has always attracted the attention of researchers. However, there are still a lot of unanswered questions regarding SVM, particularly concerning parameter selection and kernel function, which require further in-depth investigation. To enhance model performance, suitable kernel functions and parameter settings must be chosen. Although this article proposes a kernel function and parameter selection method for the SVM classification algorithm, this is only a starting point, and there are still many aspects that can be further optimized and expanded:

1. Function optimization: Numerous other kinds of kernel functions can be researched and contrasted in addition to the four kernel functions discussed in this article. We can keep looking at new options regarding kernel function selection because different problems might call for different kinds of kernel functions.

2. Enhancement of operation efficiency: variations in kernel functions can result in variations in model training timeframes. Optimizing the model's operating efficiency is a crucial objective. Future studies can concentrate on cutting training time, mainly when working with big amounts of data.

3. Parameter optimization approaches: More advanced parameter optimization techniques, such as genetic algorithms, etc., can be explored in addition to the hyperparameter search methods covered in this article. More intelligent parameter space exploration is possible with these techniques.

4. Cross-validation and network search: In practical applications, cross-validation and network search methods can help determine the optimal parameter configuration. More detailed studies could include different cross-validation strategies and search algorithms to obtain more robust results.

5. Simplification of the algorithm: Although SVM is a powerful algorithm, its complexity may limit its use in certain applications. Future research can explore how to simplify

SVM while maintaining its performance.

In general, the research on SVM classification algorithms is, for the most part, a continuously developing field with fascinating research opportunities and challenges ahead of us. I believe that through additional investigation and refinement, SVM will continue to realize its potential in various areas and provide more powerful tools for solving practical problems.

REFERENCES

[1] Xuemei Hou. A multi-class classification method for support vector machine applied to a noise-robust speech recognition. *Journal of Xi'an University of Post and Telecommunications*, 14(5):100–102, 2009.

[2] Guohe Feng. Parameter optimizing for support vector machines classification. *Jisuanji Gongcheng yu Yingyong(Computer Engineering and Applications)*, 47(3), 2011.

[3] Mathias M Adankon and Mohamed Cheriet. Model selection for the ls-svm. application to handwriting recognition. *Pattern Recognition*, 42(12):3264–3270, 2009.

[4] Minna Dou and Xiaoxia Wang. Study on optimization of process parameters based on svm. *Automation Instrumentation*, (07):67–73, 2022.

[5] Lida Wang. *Research and Application of SVM Based on the Mixed-kernel Function*. PhD thesis, Dalian: Dalian Maritime University, 2016.

[6] Sayed Fadel, Said Ghoniemy, Mohamed Abdallah, Hussein Abu Sorra, Amira Ashour, and Asif Ansary. Investigating the effect of different kernel functions on the performance of svm for recognizing arabic characters. *International Journal of Advanced Computer Science and Applications*, 7(1), 2016.

[7] Parveen Kumar, Nitin Sharma, and Arun Rana. Handwritten character recognition using different kernel based svm classifier and mlp neural network (a comparison). *International Journal of Computer Applications*, 53(11), 2012.

[8] Zhiliang Mao, Chunbo Liu, and Feng Pan. Parameter selection and application of svm with mixture kernels based on ipso. *Jiangnan University(Natural Science Edition)*, 8(6):631–634, 2009.

[9] Gend Lal Prajapati and Arti Patle. On performing classification using svm with radial basis and polynomial kernel functions. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 512–515. IEEE, 2010.

[10] Qi Zhou. A comparative study of several common kernel functions and parameter selection for support vector machines. *Fujian Computer*, (6):42–43, 2009.

[11] Shunjie Han, Cao Qubo, and Han Meng. Parameter selection in svm with rbf kernel function. In *World Automation Congress 2012*, pages 1–4. IEEE, 2012.

[12] Xiao Wu, Yan Wei, and Xia Wu. Support vector machine based on hybrid kernel function. *Chongqing University of Technology(Natural Science*, (10):66–70, 2011.

[13] Chunxi Dong, Xian Rao, Shaoquan Yang, and Songtao Xu. Method for selecting the parameters of support vector machines. *Systems Engineering and Electronics*, 26(8):1117–1120, 2004.

[14] Guangzhi Rong, Kaiwei Li, Yulin Su, Zhijun Tong, Xingpeng Liu, Jiquan Zhang, Yichen Zhang, and Tiantao Li. Comparison of tree-structured parzen estimator optimization in three typical neural network models for landslide susceptibility assessment. *Remote Sensing*, 13(22):4694, 2021.

[15] P VASANTHANAGESWARI. Improving svm classifier model using tree structured parzen estimator optimization for crop prediction. *Journal of Theoretical and Applied Information Technology*, 100(22):6808–6818, 2022.

[16] R. A. Fisher. Iris. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C56C76.