WENZHOU-KEAN UNIVERSITY

Lecture Review

CPS4982: Advanced Machine Learning

Project Name: SVM Lecture Review

Student Name:	Chun Chen (Dave)
Student ID:	1234930
Lecture:	Prof. Shitong Wang

Summer 2025

Category:

1. Introduction

Machine learning has long been considered an important tool in data analysis. However, with the explosive growth of data generation in the late 20th century, researchers have put forward higher demands for models that can handle large-scale and complex data sets. Support vector machines (SVMs) have gradually become one of the mainstream classification methods at that time due to their good generalization ability and excellent classification performance in high-dimensional space.(Guido et al., 2024)

This article aims to **review** the basic principles of SVM based on the **course content** and **explore** its adaptability and processing methods when facing different data characteristics in combination with **relevant literature**. The structure of the article will include the analysis of the standard SVM formula, the strategies adopted by SVM when dealing with specific types of data sets (such as single-class data, imbalanced data, and multi-classification problems), and finally discuss the limitations of SVM and its potential value in the era of deep learning.

2. Background and Motivation

2.1 Background

In the 20th century, before deep learning emerged, data analysis mainly relied on manually designed feature extraction methods.(Wang et al., 2019) Researchers usually construct feature functions based on domain knowledge to extract low-level information such as edges, textures, and frequencies from raw data. Such manual features often have the characteristics of high dimensionality, small sample size, and imbalanced categories. Traditional machine learning models often show insufficient generalization ability when dealing with these complex structures.

2.2 Motivation

The introduction of the Support Vector Machine (SVM) effectively addresses the challenges above. On the one hand, the optimization problem of SVM is a convex function, and its training process can ensure the **global optimal solution** and avoid falling into local minima; on the other hand, its core idea is to maximize the category interval and find the optimal linear hyperplane in the feature space to achieve efficient classification.

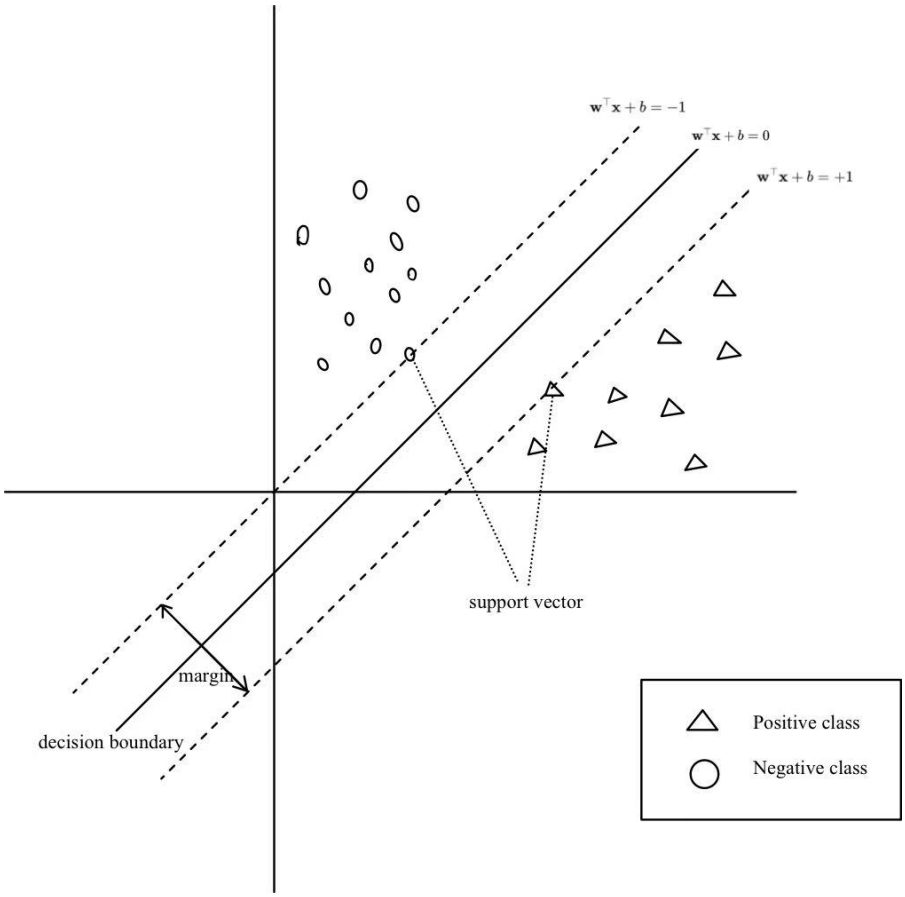
When faced with **linearly inseparable data**, SVM introduces kernel function technology to map the input samples to a higher-dimensional or even infinite-dimensional feature space, making the originally inseparable problem linearly separable in the high-dimensional space, and there is no need to explicitly construct the high-dimensional representation, thereby maintaining computational efficiency.

In addition, the **soft margin mechanism** used by SVM allows a small number of samples to be misclassified, which further improves the robustness of the model under noise interference and complex data distribution while ensuring classification accuracy.(Cervantes et al., 2020)

3. Theoretical Foundation of SVM

3.1 Linear SVM and Maximum Margin

The essence of SVM is to find a straight line (decision boundary) between two classes that can completely separate the two classes. (See Figure 1) The decision boundary can satisfy the requirement that the two vectors (support vectors) closest to it from the two classes have the longest vertical distance to the decision boundary. In SVM, we call the sum of these two vertical distances 'margin', so the task of SVM is to **find the largest margin** to separate the two classes.



(Figure 1: Basic SVM in two-dimensional space)

The equation of the decision boundary is:

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

We call the plane where the two support vectors are located the support plane, and its formula is:

$$\mathbf{w}^\top \mathbf{x} + b = \pm 1$$

And the distance between the two support planes is the ‘margin’ we need to find, which is:

$$\text{Margin} = \frac{1}{\|\mathbf{w}\|} + \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

So the problem is transformed into finding the **maximum value of 2/||w||**, that is, finding the minimum value of ||w||. However, in order to facilitate the subsequent derivation work, we transform the problem into:

$$\max \frac{2}{\|\mathbf{w}\|} \quad \Leftrightarrow \quad \min \frac{1}{2}\|\mathbf{w}\|^2$$

However, in real-world datasets, the data distribution is often imperfect and may contain a certain number of outliers or noisy samples. Enforcing a strict separation of all training points can cause the resulting decision boundary to be overly influenced by these atypical instances, thereby reducing the model's generalization ability. To address this issue, Support Vector Machines (SVMs) introduce the **soft-margin mechanism**, which incorporates a penalty term into the optimization objective to account for classification errors.

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

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

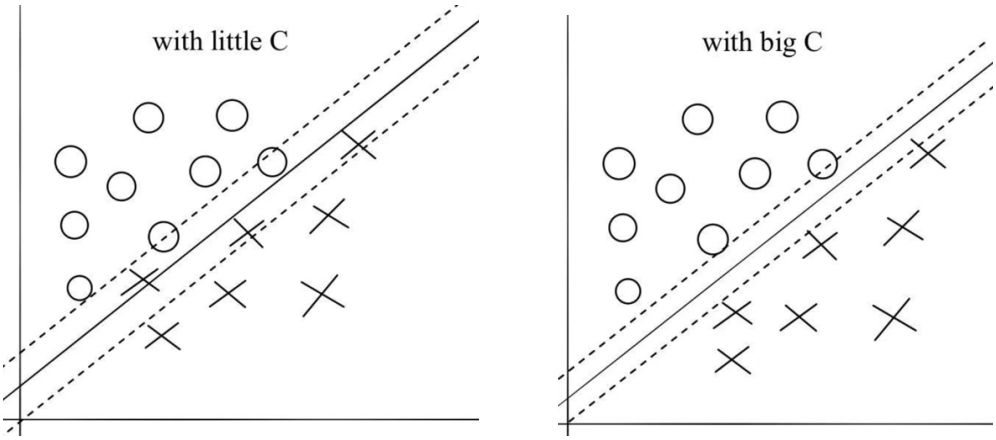
This is achieved by introducing slack variables **‘ξi’**, which quantify the degree of misclassification for each sample **‘xi’**,

2 When **ξi =0,** sample outside support border, Correct Classification.

3 **When 0 < ξi < 1** , sample between support border but on the correct side of the decision border, Correct Classification.

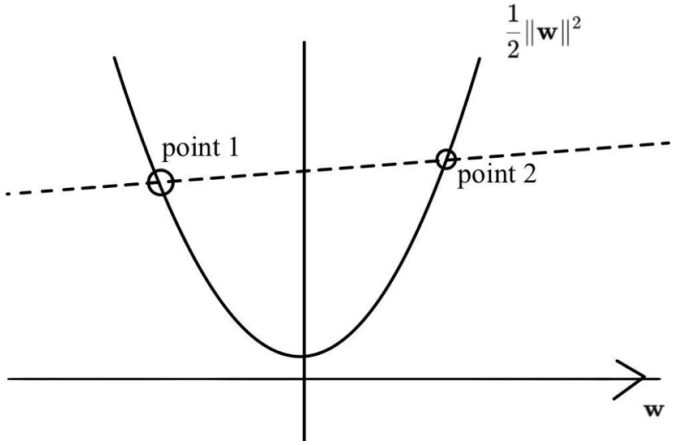
4 When **1 ≤ ξi**, the sample is on the wrong side of the decision border.

and a regularization parameter **'C'**, which controls the trade-off between maximizing the margin and minimizing the classification error.(see Figure 2) This formulation allows certain samples to be misclassified, thereby Smaller C increases slack tolerance, leading to a broader margin and a larger set of active constraints (support vectors). Larger C reduces tolerance, resulting in fewer but more critical support vectors that tightly define the separating hyperplane.



(Figure 2, with little **C,** the model is more tolerant to misclassifications and vice versa.)

Since the objective function is convex and the constraints are linear, this optimization problem is a **convex quadratic programming problem**. As a result, it guarantees a **unique global optimum** (see Figure 3), which enhances the theoretical reliability and practical stability of SVM.



$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

(Figure 3. Convexity property: Any point between point 1 and point 2 lies below the

line segment connecting them.)

So the convexity of this formulation ensures that any local minimum is also a global minimum.

In summary, we have derived the **primal form** of SVM, which formulates the task of maximizing the margin as a constrained convex optimization problem. However, due to the **presence of inequality constraints**, the problem cannot be solved directly by taking derivatives. This motivates the use of the Lagrangian dual formulation, which will be discussed in the next section.

3.2 Lagrangian Dual Formulation

To address the constrained optimization problem in the SVM primal form, we introduce two sets of Lagrange multipliers, **αi ≥ 0** and **μi ≥ 0**, corresponding to the two constraint conditions:

$$(1) \quad y_i(\mathrm{w}^\top \mathrm{xi} + b) \geq 1 - \xi_i,$$

$$(2) \quad \xi_i \geq 0.$$

These are used to construct the Lagrangian formulation of the soft-margin SVM.

When we combine the Lagrange factors and the constraints, we get the Lagrange formula in the following form:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i - \sum \alpha_i[y_i(\mathbf{w}^\top x_i + b) - 1 + \xi_i] - \sum \mu_i \xi_i$$

$$\alpha_i \geq 0 \quad , \quad \mu_i \geq 0$$

However, since the Lagrangian formulation involves five sets of variables—**w, b, ξ, α, μ**—it is not convenient to solve directly.
To simplify the optimization process, we apply the **Karush-Kuhn-Tucker (KKT)** conditions, which allow us to eliminate the primal variables **w, b, and ξi** and obtain the dual formulation

We first find the Stationarity:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad \Rightarrow \quad \mu_i = C - \alpha_i$$

$$0 \leq \alpha_i \leq C$$

Then we get a dual problem:

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C$$

After formulating the dual problem, we proceed by substituting the training data **xi** and corresponding labels **yi** into the dual objective function. We then optimize with respect to the dual variables ai , typically using a quadratic programming solver.

During this process, we observe that **many of the optimal ai values are exactly zero**, while only a subset of them satisfy 0 < **α**i < C. According to the KKT conditions, these are the **support vectors**—the training points that lie on or within the margin and have direct influence on the final decision boundary.
These **support vectors** are the only points that contribute to the final decision function:

$$f(x) = \sum_{i \in \text{SV}} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b$$

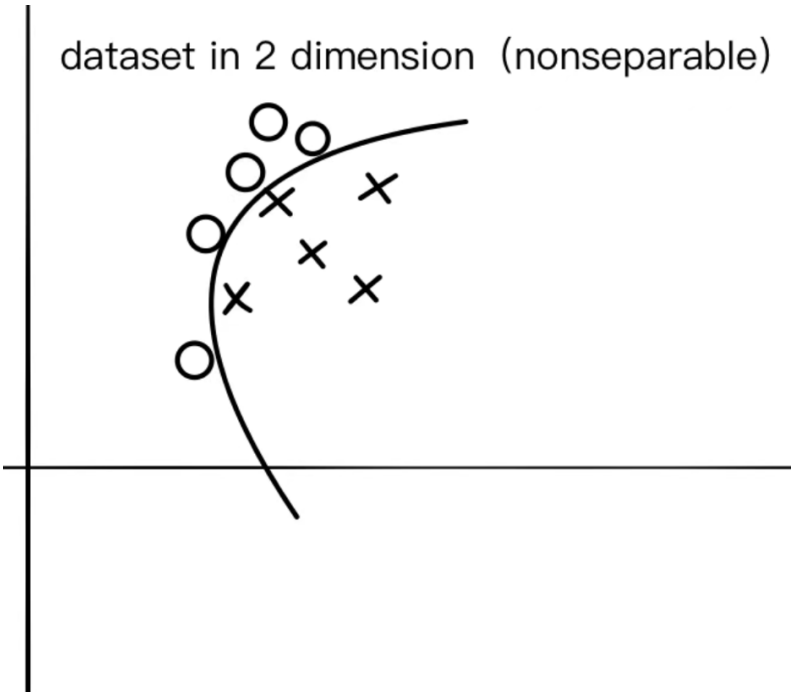
where "SV" denotes the set of indices for which αi > 0. This property of **sparsity** is one of the most powerful aspects of SVMs, making them both efficient and interpretable.

In summary, we have demonstrated how a constrained primal minimization

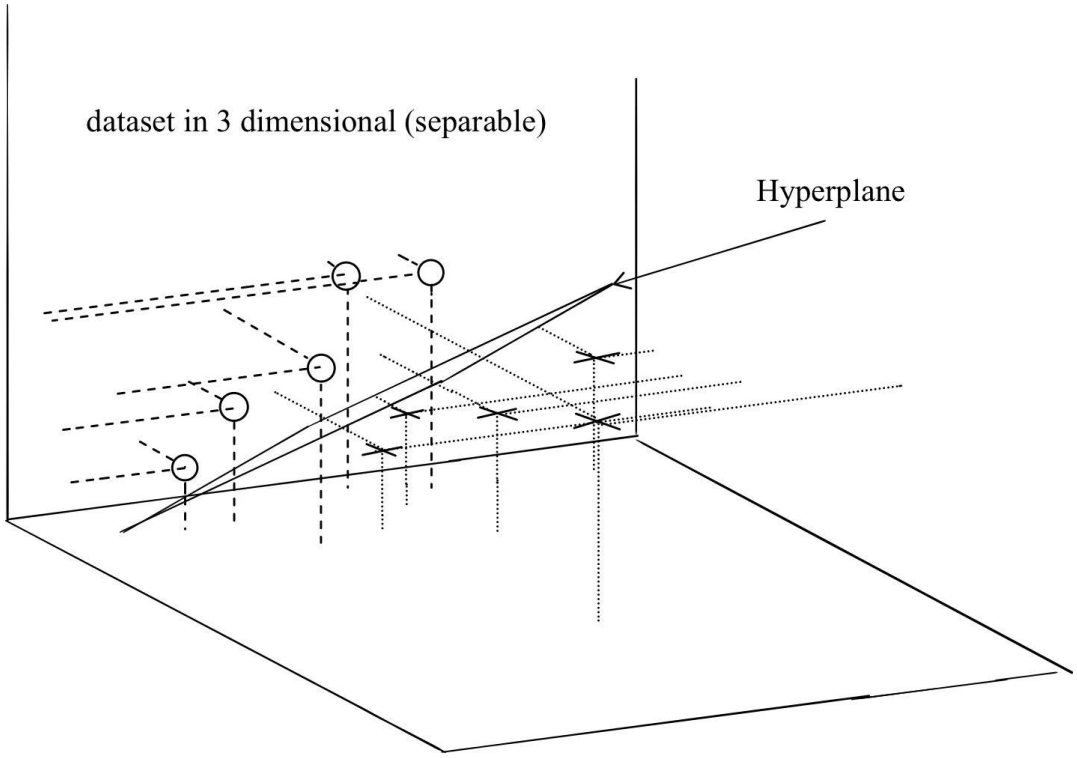
problem in SVM can be transformed into an unconstrained dual maximization problem. This transformation simplifies the optimization and enables the use of kernel functions. **The dual objective function reflects a balance: it encourages larger α values through its linear term, while penalizing redundant or conflicting support vector pairs via the quadratic interaction term.** The solution naturally emerges by maximizing this objective, guiding the model toward a **sparse** and **generalizable** decision boundary.

3.3 Kernel Trick and Nonlinear SVM

However, in real-world applications, data is often **not linearly separable** (see Figure 4) in its original feature space. To address this, SVM introduces the idea of mapping input vectors into a **higher-dimensional space** (see Figure 5), even potentially infinite-dimensional.



(Figure 4, we can not use a linear line to separate this dataset in 2-demensional)



(Figure 5, we can use a hyperplane to separate the dataset when mapping into 3-dimensional)

feature space via a nonlinear transformation, in which a linear hyperplane can effectively separate the data:

$$\mathbf{x} = [a, b] \quad \longrightarrow \quad \phi(\mathbf{x}) = [a^2,\, b^2,\, a,\, b,\, ab]$$

The above formula gives an example of converting a **two-dimensional vector into a five-dimensional vector.**

Nevertheless, explicitly computing such a high-dimensional mapping ϕ(x) and operating within that space is computationally expensive and often impractical. To overcome this challenge, SVM utilizes **kernel functions**:

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

which allows the model to compute the inner product of mapped vectors implicitly, without ever needing to compute the mapping explicitly. This approach is known as the **kernel trick**, and it forms the foundation of kernel-based SVMs.

Based on the principle of kernel function, we can reasonably replace the inner product of the **support vector** and **input vector** in the dual function or decision boundary with their corresponding **kernel function**:

dual function use kernel function:

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

decision boundary uses kernel function:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

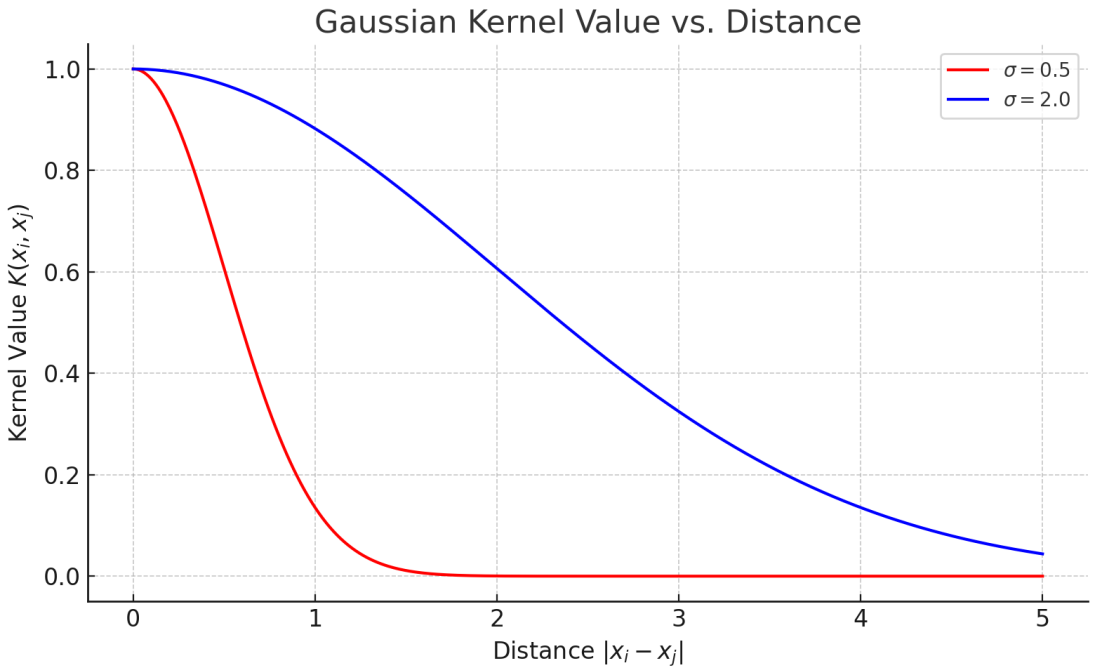
In general, the introduction of kernel functions allows support vector machines to calculate equivalent inner products through kernel functions without explicitly constructing high-dimensional mappings ϕ(x) when dealing with nonlinear problems, thereby achieving linear classification in high-dimensional feature spaces.

However, kernel functions still have **limitations in some applications**. For example, different types of kernel functions may show significant differences on the same data set, and model performance is highly sensitive to kernel function parameters.(see Figure 6)

A typical example is the **Gaussian kernel function**:

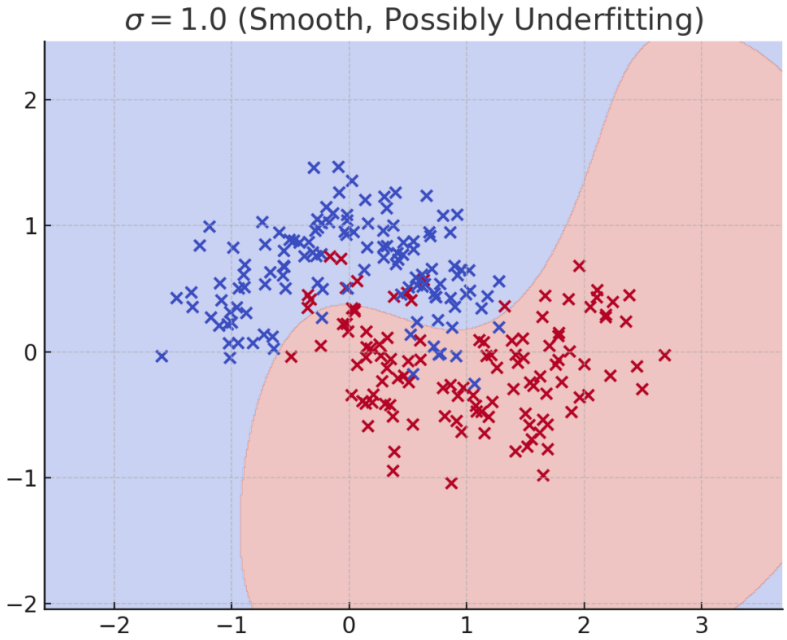
$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

The kernel function parameter **σ** directly determines the calculation method of similarity between samples, which has a profound impact on the selection of support vectors, the shape of decision boundaries, and the generalization ability of the model. **When σ is large**, the decay rate of the kernel function slows down, resulting in most sample pairs being judged as highly similar, and the model is difficult to accurately distinguish points near the boundary from those far away from the boundary; **when σ is small**, only samples that are close enough are considered similar, making the model more sensitive to local structure.(see Figure 6)



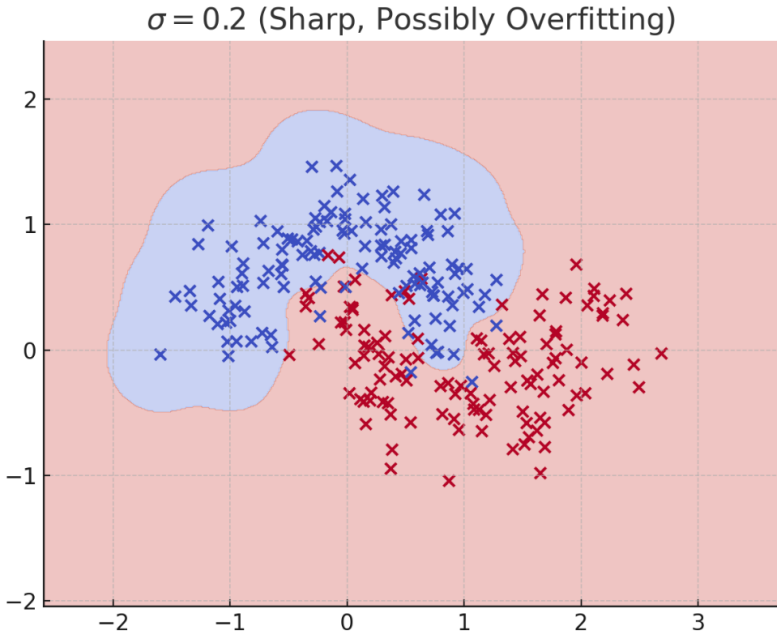
(Figure 6, The kernel function is insensitive to large **σ** changes in distance, but sensitive to small **σ**)

At the local level, SVM optimizes the α of similar samples to avoid multiple highly similar points from participating in the construction of the decision boundary at the same time, thereby suppressing redundancy and reducing the local penalty term in the dual function; but at the global level, the model still needs to select sample points with reasonable distribution and non-redundant information from the input space to maintain the stability of the boundary. Therefore, **when σ is large**, the model will rely on more support vectors due to the characteristic of "blurring all samples". Even if the model keeps a single support vector α small, the whole model still shows a wide response to the global structure, which may lead to **underfitting**.(see Figure 7)

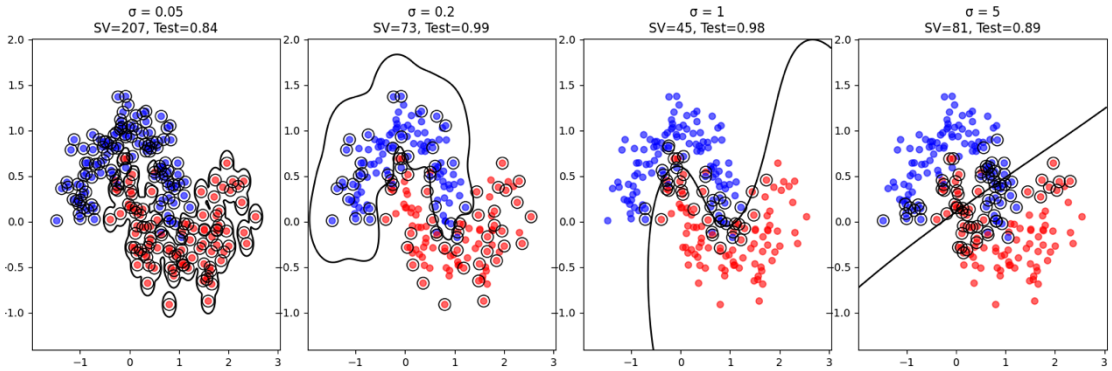


(Figure 7, when we change the dimension from high dimension back to two dimensions, we can see underfitting when the model responds broadly to all vectors)

When **σ** is small, the kernel function sharply decays, making the model highly sensitive to local variations. The decision boundary relies heavily on only a few nearby support vectors. If any of these are outliers or atypical samples, the model may overfit by shaping the boundary around them, leading to **poor generalization**.(see Figure 8)



(Figure 8, when we change the dimension from high dimension back to two dimensions, we can see overfitting When the model over-responds to some vectors)



(Figure 9, different σ with different support vector numbers and test accuracy)

Looking at another set of data, when σ becomes extremely small, the influence of each support vector is limited to a very narrow local region. This leads to highly localized curvature of the decision boundary, which may appear fragmented visually, corresponding to high model capacity and potential overfitting.

Conversely, when σ becomes extremely small, the RBF kernel function approaches a nearly constant similarity metric between samples. In this case, the model effectively degenerates into a linear classifier. For inherently non-linear datasets, this reduction in capacity can lead to boundary violations, causing many samples to reach the upper bound ($\alpha_i = C$). These samples are not "mistakenly" assigned as support vectors; rather, they become effective constraints due to underfitting.

In summary, the kernel function parameters essentially control the model's perception of **local differences and global structures**, and are the key factors affecting the balance between SVM **classification accuracy, sparsity and generalization ability**. In addition, kernel methods often have high computational overhead when facing large-scale data, especially when constructing and storing kernel matrices, which limits their practical application in the context of big data.(Kuyoru et al., 2020),(Metos et al., 2016)

4. Extensions of SVM

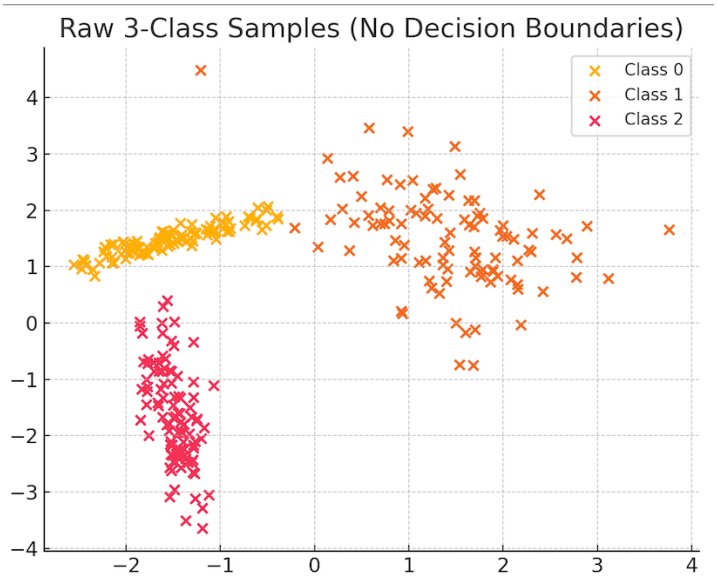
In the previous article, we reviewed how support vector machines transform constrained optimization problems into solvable maximization problems through **duality**, and discussed how the model constructs an ideal decision boundary through **nonlinear mapping**, **kernel function techniques**, and **parameter adjustment**.

However, in practical applications, data distribution is often far from ideal, and there may be label imbalance, multi-category classification, or even lack of labels.

This section will summarize the extension methods and coping strategies of the support vector machine introduced **in class** when facing these **complex data conditions**.

4.1 Multi-Class SVM

In practical applications, samples often involve multiple category labels rather than just binary classification,(see Figure 10) which makes it impossible for traditional SVM to directly define a clear multi-class decision boundary.



(Figure 10, 3 labels in this dataset)

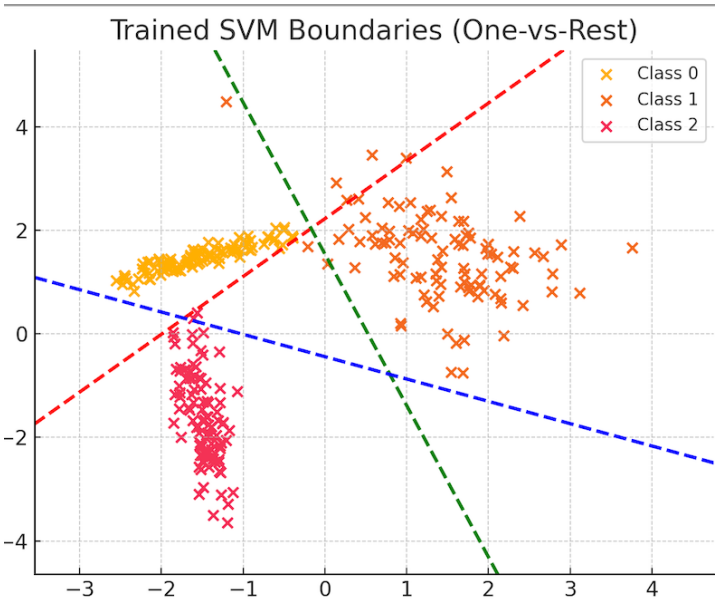
To solve this problem, the One-vs-Rest (OvR) strategy converts multi-

classification tasks into multiple binary classification problems. The basic principle is: each **label is regarded as a positive class in turn**, and all other labels are combined into **negative classes**. A **binary classifier** is trained using a support vector machine to learn a decision boundary that separates this class from other classes.

In the whole process, if there are k labels in total, k independent SVM models will be trained, each corresponding to a label, and a score function will be output. The score function is essentially a standard SVM decision function:

$$f_k(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^{(k)} y_i^{(k)} K(\mathbf{x}_i, \mathbf{x}) + b^{(k)}$$

whose value distinguishes the current label (positive class) from all other labels (negative class) (see Figure 11), and the score function is defined by $fi(x)$=0, that is, the model's confidence in the sample belonging to this class is at the critical point of judgment

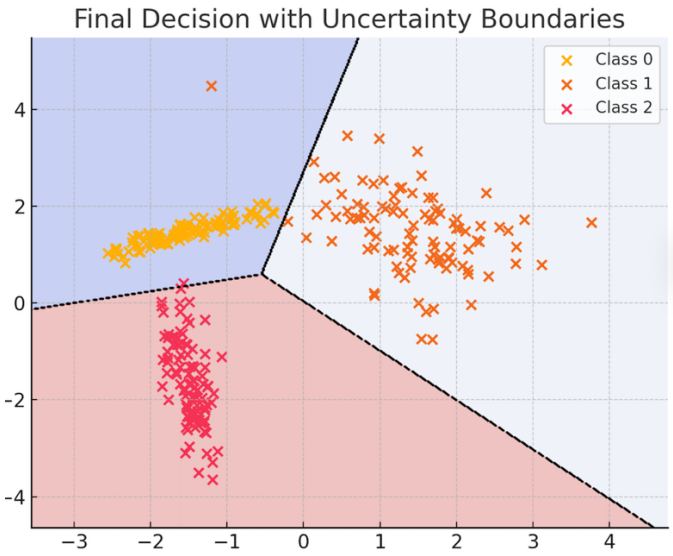


(Figure 11, each label has its own score function of the SVM model to separate from the other two labels)

For any input sample, all models calculate their scores separately, and the final prediction result depends on which category has the largest output value of the score function. When a sample is exactly at a position where the scores of two categories are equal:

$$fi(x) = fj(x)$$

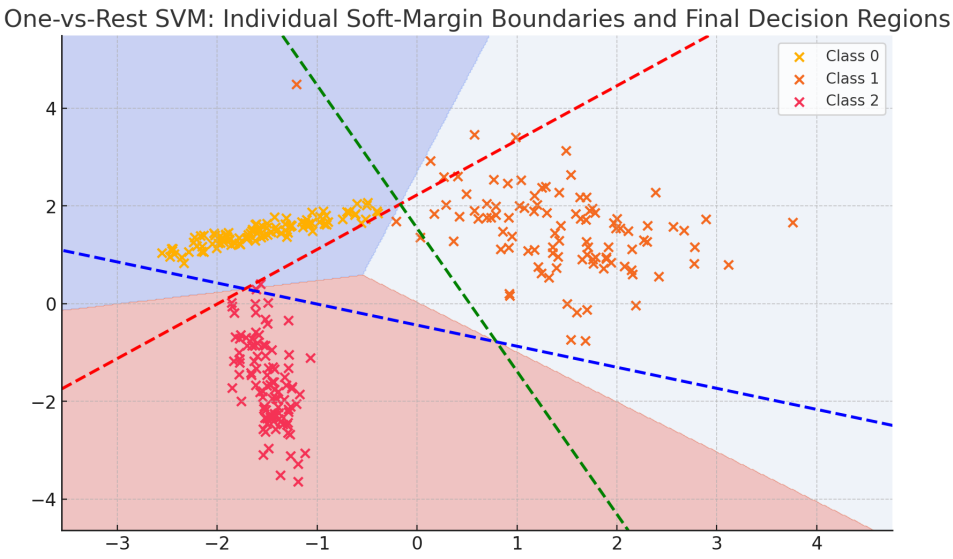
The sample is on the "classification boundary" between the two. This kind of boundary can also be understood as the "uncertainty boundary" of the model, that is, the decision boundary when the model cannot clearly determine the label between multiple classifiers.(see Figure 12)



(Figure 12, The black lines indicate where the two SVM classifiers have equal scores)

In general, this type of multi-class SVM generates multiple scoring functions corresponding to each class by building an SVM model for **each label** that can distinguish it from **all other labels**. Although each scoring function itself corresponds to a standard binary decision boundary, in the entire multi-classification task, the final classification result is not determined by a single boundary, but by the maximum value **comparison between multiple scoring functions.**

Therefore, the overall classification boundary is jointly constituted by the "**relative advantage relationship**" between these scoring functions, rather than the independent judgment results of each. See the difference between scoring functions and classification boundary (see Figure 13)



(Figure 13, the dotted line is the score function determined by the SVM model, and the solid line is the classification boundary obtained by the score function through the relative relationship of the scores)

4.2 One-Class SVM

Unlike multi-class SVM, One-Class SVM does not rely on existing labels to classify different categories.

It learns the overall distribution structure of normal samples and builds a boundary, which is a circle with radius ***R***, that surrounds most data points as much as possible. The **distances** from all vectors in it to the center of the circle ***c*** must satisfy the **constraints**. Therefore, we can get a primal SVM model based on this:

$$\min_{R,c} \quad R$$

$$\text{subject to} \quad (x_i - c)^\top (x_i - c) \leq R, \quad \forall i = 1, 2, ..., n$$

This method only uses normal samples for training and does not rely on the label information of abnormal samples. The basis for whether the test sample is normal is by judging whether the sample falls outside the boundary.

However, in practice it is necessary to add bias redundancy to the model, so we introduce slack variables '**ξi**' into the model:

$$\min_{R,c,\xi} \quad R + C\sum_{i=1}^{n}\xi_i$$

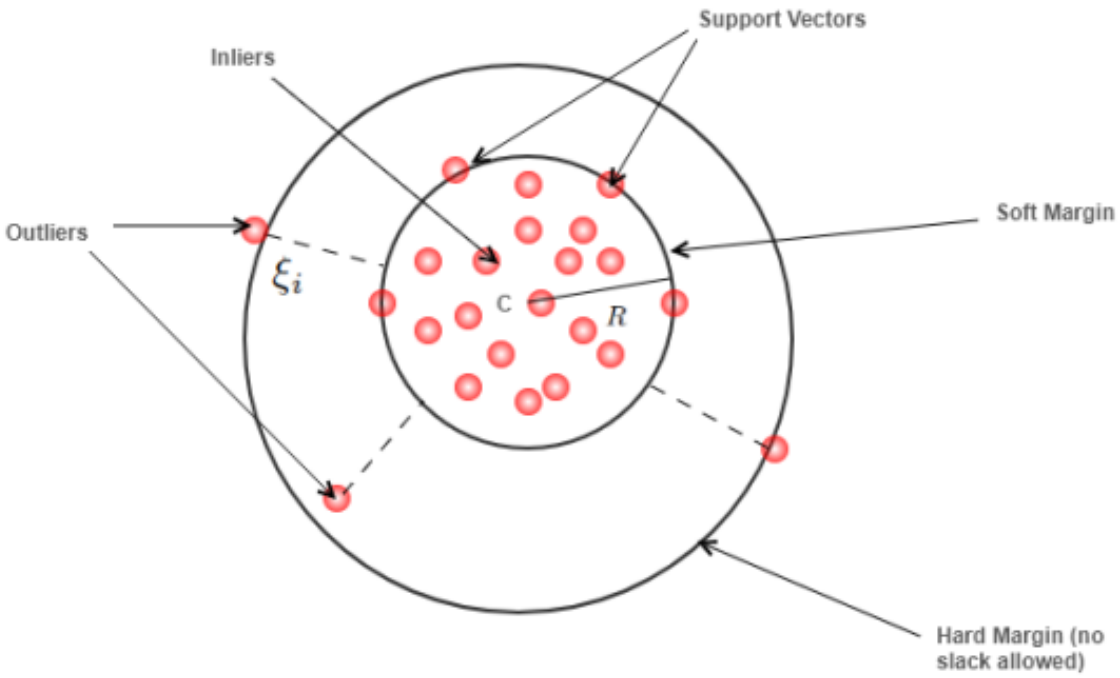
$$\text{subject to} \quad (x_i - c)^\top (x_i - c) \le R + \xi_i, \quad \xi_i \ge 0, \quad \forall i$$

After getting primal SVM, we need to introduce the Lagrange multiplier ***'αi'*** and ***'μi'*** combined with two constraints, Next, we find the partial derivatives of the variables in the Lagrange model and finally simplify it to a decision boundary with only α:

$$\mathcal{L}_D(\alpha) = \sum_{i=1}^{n}\alpha_i\|x_i\|^2 - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j x_i^\top x_j$$

$$\text{subject to} \quad \sum_{i=1}^{n}\alpha_i = 1, \quad 0 \le \alpha_i \le C$$

Finally We can get a one-class SVM model with soft margin (see Figure 14)



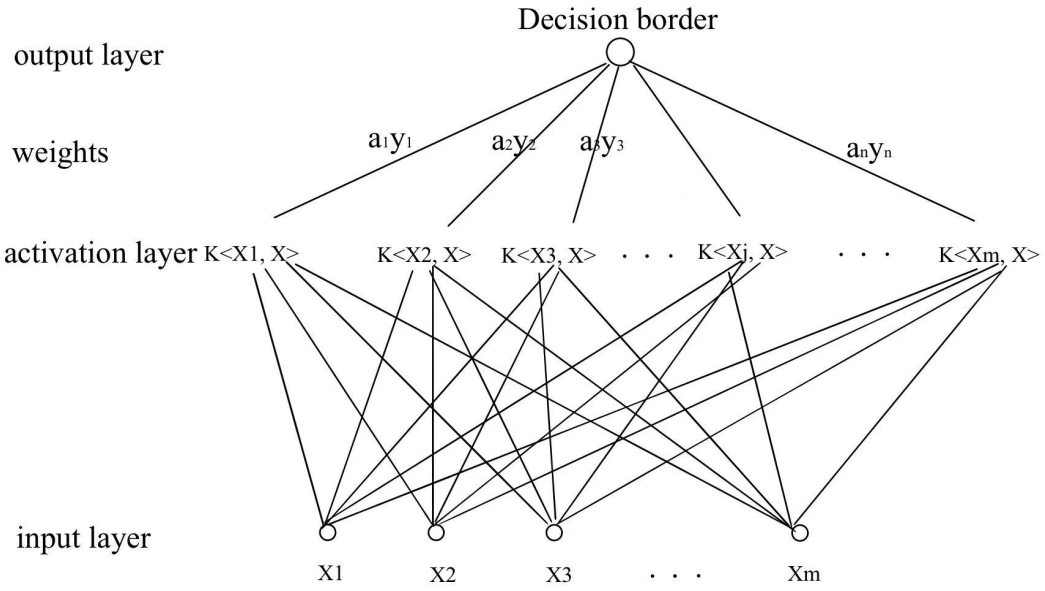
(Figure 14, one-class SVM with soft margin)

Unlike binary SVMs that aim to maximize the margin between two labeled yi = -yj classes, One-Class SVM is designed to learn a tight boundary that encloses the majority of a single-class dataset. While binary SVM requires both positive and negative examples, One-Class SVM only needs data from one “normal” class and treats points outside the boundary as anomalies.

5. Visualization and Interpretation

In order to better understand the SVM process, we can decompose the training structure of SVM into a three-layer structure of a neural network. (see Figure 15)

- The first is the input layer, in which we need to input the **training vectors** in sequence.
- The second is the activation layer, in which we use the **kernel function** to calculate the similarity between the input vector and all support vectors. In the initial state, all vectors are regarded as support vectors, that is, when we have k input vectors, the initial support vectors are also k.
- The **weight** is between the second and third layers, which is composed of the Lagrangian multiplier ***'αi'*** and the label corresponding to each support vector.
- The third layer is the output layer. When all training vectors are input, the output layer can classify any new input. In essence, the output layer is the **decision boundary** of SVM.



(Figure 15, SVM model of the neural network structure in the initial state)

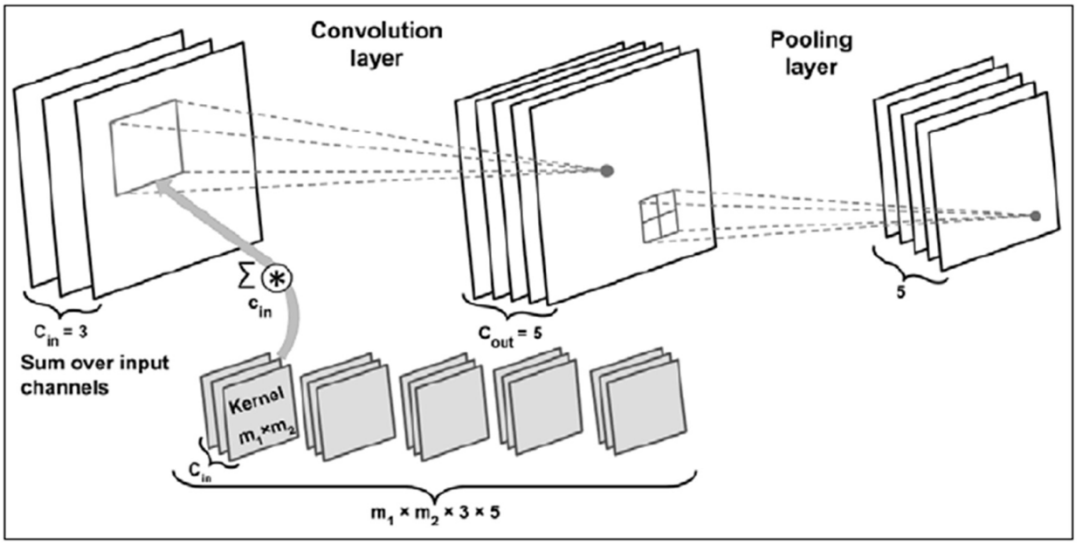
After all training vectors are inputted in sequence, the model continuously iterates the kernel function and the Lagrange multiplier ***'αi'*** corresponding to each vector, and finally only retains the training vectors corresponding to **0<*αi*<C** as **support vectors** for use in the kernel function of the final model. Other vectors have **no effect on the decision boundary** of the model because ***αi*=0**, so

they are removed from the model.

6. Applications from Recent Research Papers

In recent years, although **deep learning and artificial intelligence technologies** have made breakthrough progress in many tasks and have gradually replaced the dominant position of traditional models in processing high-dimensional complex data, support vector machines (SVMs) still play an important role in many key areas, especially for high-dimensional and small sample application scenarios.

- In medical diagnosis, SVMs are widely used in tasks such as cancer detection, gene expression data analysis, and disease classification. They are often used in combination with feature selection techniques such as PCA and LASSO to effectively deal with the dimensionality disaster problem and improve model stability.(Guido et al., 2024)
- In the field of image recognition, SVMs are often used as an alternative to the output layer of deep convolutional neural networks (CNNs) (see Figure 16), especially for small sample or fine-grained image classification tasks, such as facial expression recognition and remote sensing image analysis. By inputting deep features into the SVM classifier, it is possible to balance the extraction capability and the stability of the decision boundary. (Haider et al., 2023; Kowsari et al., 2019)



(Figure 16, SVM takes the output value of the previous layer, such as the feature vector extracted by the fitter, and inputs it into the next layer)

- In natural language processing (NLP) tasks, SVM is used in practical applications such as sentiment analysis, spam identification, and text

classification, and is often combined with TF-IDF, Word2Vec, or BERT embedding vectors to build efficient and interpretable text classification models, especially for small and medium-sized corpus scenarios.(Cervantes et al., 2020)

Overall, SVM still occupies a place in modern machine learning systems with its good generalization ability, clear theoretical basis, and strong interpretability. Especially when combined with modern feature extraction methods (such as deep learning models), SVM is often used as a robust backend classifier, showing good complementary value and application prospects in various tasks.

7. Limitations and Discussion

Although support vector machines (SVMs) show good robustness in processing high-dimensional feature spaces and unknown data, they still have some significant limitations.

First, SVM is a model that is **highly sensitive to parameters**. Improper settings of hyperparameters such as the choice of kernel function, regularization parameter **'C'**, and kernel parameter **'γ'** can easily lead to **overfitting or underfitting** of the model.

Second, SVMs have a certain dependence on the distribution of input data. In the case of uneven distribution of labels, the model tends to be more biased towards the **majority class**, thus affecting the recognition effect of the **minority class**.

Although deep learning technology has achieved leading performance in many tasks, SVMs still have their unique advantages. In particular, in scenarios with limited sample size or high interpretability requirements, SVMs still have application value due to their theoretical controllability and boundary construction mechanism. Research in recent years has also explored hybrid architectures that combine SVMs with deep learning, such as using SVMs as output classifiers for models such as CNN and BERT to improve classification performance and generalization capabilities.(Karamizadeh et al., 2014)

8. Conclusion

This paper systematically reviews the basic principles, mathematical derivation process, and various extension forms of support vector machines (SVMs), including One-vs-Rest multi-classification SVMs and One-Class SVMs. We analyze in detail how they adapt to different types of data and task scenarios, and combine visualizations to demonstrate the construction logic of their decision boundaries.

Although SVM is no longer a mainstream architecture in the context of the prevalence of deep learning, it still has irreplaceable value in specific fields (such as medical images, small sample classification, text analysis, and anomaly detection). In particular, when combined with modern feature extraction methods (such as CNN and BERT), SVM can be used as a stable and efficient backend classifier to improve the interpretability and generalization ability of the overall model.

In the future, the development of SVM can continue to move in the following directions: kernel function adaptive optimization, large-scale approximate training, integration with deep models, online/incremental learning, etc. In these directions, SVM is expected to continue to expand its application boundaries in modern machine learning systems.

9. References

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., & Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, *408*, 189–215. https://doi.org/10.1016/j.neucom.2019.10.118

Guido, R., Ferrisi, S., Lofaro, D., & Conforti, D. (2024). An Overview on the Advancements of Support Vector Machine Models in Healthcare Applications: A Review. *Information*, *15*(4), 235. https://doi.org/10.3390/info15040235

Haider, I., Yang, H.-J., Lee, G.-S., & Kim, S.-H. (2023). Robust Human Face Emotion Classification Using Triplet-Loss-Based Deep CNN Features and SVM. *Sensors*, *23*(10), 4770. https://doi.org/10.3390/s23104770

Karamizadeh, S., Abdullah, S. M., Halimi, M., Shayan, J., & Rajabi, M. J. (2014). Advantage and drawback of support vector machine functionality. *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, 63–65. https://doi.org/10.1109/i4ct.2014.6914146

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E., & Brown, D. E. (2019). Text Classification Algorithms: A Survey. *Information*, *10*(4), 150. https://doi.org/10.3390/info10040150

Kuyoro, S. O., Sulaimon, O. R., & Ojewande, O. O. (2022). Comparative analysis of the performance of various support vector machine kernels. In *5th International Conference on Information Technology, Education and*

Development (ITED). Retrieved from https://www.researchgate.net/publication/365365650

Matos, S., & Raghava, G. P. S. (2016). Kernel-based SVM. In *Support Vector Machines Applications* (pp. 63–71). Springer. https://doi.org/10.1007/978-3-319-41063-0_5