# Linear Algebra and Its Applications Term Project

**Hung-Ta, Wang**
Institute of Industrial Engineering
National Taiwan University
r13546017@ntu.edu.tw

## 1 Rationale

### 1.1 Introduction

Machine learning has become more and more popular nowadays. It can not only fit a number of nonlinear problems but also performs with great accuracy and adaptability in various situations we encounter.

Among these machine learning methods, artificial neural networks have developed rapidly in recent years, however, these advanced methods still have some limitations. I summarized some demerits of artificial neural networks:

1. Artificial neural networks are highly dependent on enormous amounts of data, and the training process costs a lot in both time and computing resources.
2. Since we apply a lot of activation functions and layers in the model, the model's explainability is poor. Therefore, it is difficult to understand the training process and improve the model.
3. When the model demonstrates high prediction accuracy, it is necessary to determine whether this performance stems from the model's effectiveness or is merely a result of overfitting.

In conclusion, though artificial neural networks are powerful, we cannot neglect the importance of other traditional methods like Support Vector Machine (SVM) that are more mathematically based, explainable, and robust.

### 1.2 Motivation

SVM performs excellently in many fields, including "Speech emotion recognition using support vector machine"[2] or "Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review"[5].

Most modern tasks like emotion recognition or image classification are completed by Natural Language Processing (NLP) or Convolutional Neural Networks (CNN), however, SVM should not be a forgotten method.

SVM can demonstrate better results than other modern methods in some situations. Especially in small-sample or high-dimension circumstances, SVM is a more robust method[4]. Even if we cannot certify training results, we can still adjust the model parameters to optimize the results through its high explainability by the solid mathematical foundation of SVM.

To sum up, based on its practicality and structure in linear algebra. I regard SVM as suitable as a topic for this term project.

## 2 Problem background

### 2.1 Parameters

Table 1: Parameters in this term project

| Parameter | Description |
|---|---|
| $m$ | A constant number of data points |
| $n$ | A constant number of features |
| $W_{m\times 1}$ | The weight vector that is perpendicular to the decision boundary |
| $w_i$ | The $i^{th}$ weight in $W_{m\times 1}$ |
| $X_{m\times n}$ | The matrix of $m$ data points with $n$ features |
| $x_{ij}$ | The data of the $j^{th}$ feature of the $i^{th}$ data point in $X_{m\times n}$ |
| $Y_{m\times 1}$ | The vector of types of all data points |
| $y_i$ | The type of the $i^{th}$ data point |
| $b$ | The bias that also stands for the interception of the hyperplane |
| $\xi_i$ | The distance that the $i^{th}$ data point is classified in the wrong type |
| $C$ | A constant of the tolerance of the error |
| $l_i^{hinge}$ | The hinge loss of the $i^{th}$ data point |

### 2.2 Support Vector Machine

#### 2.2.1 Definition

I would like to introduce SVM from its definition and geometry. The description is inspired by "Support vector machines–an introduction"[4] and "Data classification using Support Vector Machine (SVM), a simplified approach"[1].

Firstly, Figure 1 below is plotted by Python, and it illustrates the fundamental concept of SVM

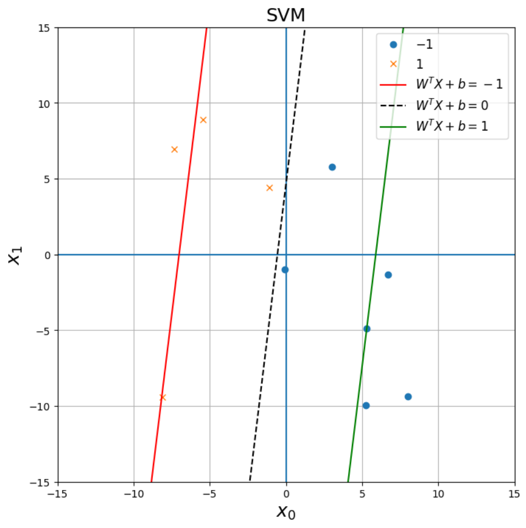


Figure 1: SVM Demostration

SVM can be divided into three parts:

- Decision boundary: $W^T x + b = 0$
- Positive class support plane: $W^T x + b = 1$
- Negative class support plane: $W^T x + b = -1$

The relationship between these three hyperplanes is that the decision boundary is the classification baseline. Thus, it is in the middle of the other two hyperplanes, and the distance from the decision boundary to the support plane is called the margin. We will show that both margins have the same value of distance below.

### 2.2.2 Primal Form

The margin from the decision boundary to the positive class support plane is

$$\frac{\mid (W^TX+b)_{decision\ boundary} - (W^TX+b)_{positive\ class\ support\ plane} \mid}{||W||}$$
$$= \frac{\mid 0-1 \mid}{||W||} = \frac{1}{||W||}$$

The margin from the decision boundary to the negative class support plane is

$$\frac{\mid (W^TX+b)_{decision\ boundary} - (W^TX+b)_{negative\ class\ support\ plane} \mid}{||W||}$$
$$= \frac{\mid 0-(-1) \mid}{||W||} = \frac{1}{||W||}$$

Our objective is to maximize the margin between the two classes

$$maximize \quad \frac{2}{||W||}$$

Since it is difficult to optimize when $||W||$ is the denominator, we transform the problem into an minimization

$$\begin{aligned} &minimize \quad \frac{||W||}{2} = \frac{1}{2}W^TW \\ &subject\ to \\ &\qquad y_i[W^TX+b] \geq 1 \end{aligned}$$

### 2.2.3 Dual Form

Since we define the primal form, it can be transformed into the dual form that provides additional properties to the primal form.

The primal form of the problem

$$\begin{aligned} &minimize \quad \frac{1}{2}W^TW \\ &subject\ to \\ &\qquad y_i[W^Tx_i+b] \geq 1 \end{aligned}$$

Then we can use Lagrange multipliers ($\alpha_i$) to reform the primal form

$$\begin{aligned} &minimize \quad L = \frac{1}{2}W^TW - \sum_{i=1}^{m} \alpha_i(y_i[W^Tx_i+b]-1) \\ &\alpha_i \geq 0 \end{aligned}$$

Since our objective is to maximize $\alpha_i$, we have to find the saddle points of $W$ and $b$ by KKT condition.

$$\frac{\partial L}{\partial W} = 0 \to W = \sum_{i=1}^{m} \alpha_i y_i x_i$$
$$\frac{\partial L}{\partial b} = 0 \to b = \sum_{i=1}^{m} \alpha_i y_i = 0$$

Recalculate $\frac{1}{2}W^T W$ with new W

$$\begin{aligned}\frac{1}{2}W^T W &= \frac{1}{2}(\sum_{i=1}^{m} \alpha_i y_i x_i)^T (\sum_{j=1}^{m} \alpha_j y_j x_j) \\ &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j\end{aligned}$$

We can reform the dual form $L_\alpha$

$$\begin{aligned}&minimize \quad L_\alpha = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ &subject\ to \\ &\qquad \sum_{i=1}^{m} \alpha_i y_i = 0\end{aligned}$$

Define new notations to reform the dual form

- $\alpha = [\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_m]^T$
- $f = [1, 1, 1, \cdots, 1]^T$
- $H = [y_i y_j x_i^T x_j]$

New dual form

$$\begin{aligned}&minimize \quad L_\alpha = -0.5\alpha^T H \alpha + f^T \alpha \\ &subject\ to \\ &\qquad Y^T \alpha = 0 \\ &\qquad \alpha_i \geq 0\end{aligned}$$

### 2.2.4 Model Generalization

To achieve the generalization of SVM, we add the concept of error tolerance. Thus the model is revised into the form (We use primal form to be example here)

$$\begin{aligned}&minimize \quad \frac{1}{2}W^T W + C\sum_{i=1}^{m} \xi_i \\ &subject\ to \\ &\qquad y_i[W^T X + b] \geq 1 - \xi_i \\ &\qquad \forall i \in [1, m],\ \xi_i \geq 0\end{aligned}$$

To simplify the problem, we substitute $\xi_i$ into hinge loss. The reason why I chose this concept is inspired by "On the error resistance of hinge-loss minimization"[6].

The hinge loss can be shown below

$$l_i^{hinge} = max(0,\ 1 - y_i W^T x_i)$$

The final model is

$$minimize \quad Z = \frac{1}{2}W^T W + C\sum_{i=1}^{m} l^{hinge}$$

#### 2.2.5 Training Process of the Final Model

After we define the model, we have to find the optimal value. Since $W$ and $b$ are the decision variables, we compute their gradient.

For gradient of $W$

- When $y_i(W^T x_i + b) < 1$:

$$\frac{\partial}{\partial W}(\frac{1}{2}||W||^2 + C\max(0, 1 - y_i(W^T x_i + b))) = W - Cy_i x_i$$

- When $y_i(W^T x_i + b) \geq 1$:

$$\frac{\partial}{\partial W}(\frac{1}{2}||W||^2 + C\max(0, 1 - y_i(W^T x_i + b))) = W - 0$$

- In total, we can get

$$\frac{\partial Z}{\partial W} = W - C\sum_{i=1}^{m} y_i x_i$$

For gradient of $b$

- When $y_i(W^T x_i + b) < 1$:

$$\frac{\partial}{\partial b}(\frac{1}{2}||W||^2 + C\max(0, 1 - y_i(W^T x_i + b))) = Cy_i$$

- When $y_i(W^T x_i + b) \geq 1$:

$$\frac{\partial}{\partial b}(\frac{1}{2}||W||^2 + C\max(0, 1 - y_i(W^T x_i + b))) = 0$$

- In total, we can get

$$\frac{\partial Z}{\partial b} = -C\sum_{i=1}^{m} y_i$$

Just like the gradient descent algorithm, we apply the learning rate to avoid local optima and prevent it from converging too slowly.

$W = W -$ learning rate $\times \frac{\partial Z}{\partial W}$

$b = b -$ learning rate $\times \frac{\partial Z}{\partial b}$

### 2.2.6 Kernel

For the merits of the dual form, we can substitute the original $x_i^T x_j$ into the kernel to make SVM adapt to the nonlinear problem. My recognition of the kernel is referred to by "Tutorial on support vector machine (SVM)"[3].

Review the new dual form

$$
\begin{aligned}
&minimize \quad L_\alpha = -0.5\alpha^T H\alpha + f^T\alpha \\
&subject\ to \\
&\qquad\quad Y^T\alpha = 0 \\
&\qquad\quad \alpha_i \geq 0
\end{aligned}
$$

$H$ is a positive definite matrix and $\alpha^T H\alpha$ can expand to $\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$, it has the following properties

- $\alpha_i\alpha_j$: The influence of interactions between the data point $i$ and the data point $j$.
- $y_i y_j$: If the data point $i$ and the data point $j$ are the same, $y_i y_j$ would be $1$. Otherwise, it would be $-1$.
- $x_i^T x_j$: The inner product of the data point $i$ and the data point $j$. It is equilibrium to $||x_i||\ ||x_j||\ cos\theta$. We can also divide the situation into three situations
  1. $x_i^T x_j > 0$: It means $\theta < 90°$, the data point $i$ and the data point $j$ have a positive correlation.
  2. $x_i^T x_j = 0$: It means $\theta = 90°$, the data point $i$ and the data point $j$ is orthogonal. They are irrelevant.
  3. $x_i^T x_j < 0$: It means $\theta > 90°$, the data point $i$ and the data point $j$ have a negative correlation.

Since $x_i^T x_j$ stands for the feature space, we can substitute it into other transformations of the space named kernel. Kernels turn the decision boundary from a straight line into a curve. Here are the kernels in the model and some examples.

$$
\begin{aligned}
&\alpha^T H\alpha = \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j = \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j K(x_i, x_j) \\
&where \quad K(x_i, x_j) = \Phi(x_i)\cdot\Phi(x_j)
\end{aligned}
$$

Types of kernels

- Linear: $K(x_i, x_j) = x_i^T x_j$, original form that the decision boundary is linear.
- Polynomial: $K(x_i, x_j) = (x_i^T x_j + c)^d$, it can capture the nonlinear rule of the data and make the decision boundary a curve by the polynomial.
- Gaussian Radial Basis Function: $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$, it can project the feature space onto an unlimited dimension hyperplane.

## 3 Solutions with linear algebra theories and techniques

### 3.1 Vector Space and Eigenvalue

#### 3.1.1 Vector Space

In the beginning, we have to define the feature space $X_{m\times n}$ of data points and the weight vector $W_{m\times 1}$.

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \cdots & x_{mn} \end{bmatrix} \qquad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

Each row of $X$ is properties of a data point and each column of $X$ is the data regarding to the feature.

- $\Re(X^T) \in \mathbb{R}^m$: the linear combination of data points
- $\Re(X) \in \mathbb{R}^n$: the linear combination of features.

Since we always have more data points than features, it is supposed $m > n$. The maximum value of Rank(X) is $n$. Although it means there are linear dependencies among data points, it weakens the influence of white noises and make the norm of the $j^{th}$ feature more close to the real value. It can help us to define the real $||X_{feature}||$.

### 3.2 Orthogonality and Projection

#### 3.2.1 Orthogonality

We can interpret $W^TX + b$ as $W \cdot X + b$. It means the inner product of $W$ and $X$ plus the intercept $b$. Since $||X_{feature}||$, $||W||$ and $||b||$ in the trained model is a certain value, the magnificence of $W \cdot X + b = ||W||\ ||X_{feature}||\ cos\theta + b$ will only be affected by $\theta$.

The positivity of $W^TX + b$ stands for the correlation between $W$ and $X$. The value $W^TX + b$ will become larger if $W$ and $X$ are more correlated. If $W^TX + b$ is close to 0, it means this hyperplane can hardly classify the type of this data point. In the worst case, $W^TX + b = 0$ means this hyperplane is not effective in classification that is the decision boundary.

#### 3.2.2 Projection

If we want to get the projection matrix directly, the function is too complicated to calculate. Let $\Phi$ be $X$ after transformation.

$$P_\Phi = \Phi(\Phi^T\Phi)^{-1}\Phi^T$$

Thus, we transform $X$ and then calculate the projection matrix. The kernel is the tool that projects the feature space onto the hyperplane with a higher dimension.

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) = \begin{bmatrix} K(x_i, x_i) & K(x_i, x_j) \\ K(x_j, x_i) & K(x_j, x_j) \end{bmatrix}$$

$$\rightarrow \quad P_\Phi = K(x_i, x_j)[K(x_i, x_j)^T K(x_i, x_j)]^{-1} K(x_i, x_j)^T$$

Where $K$ is a semi-positive definite matrix, its eigenvalues must be nonnegative.

### 3.3 Quadratic Function in Matrix Form

In the dual form of SVM

$$minimize \quad L_\alpha = -0.5\alpha^T H\alpha + f^T\alpha$$

Where $H = y_i y_j K(x_i, x_j)$ is a semi-positive definite matrix because of $K(x_i, x_j)$. We can see $y_i y_j$ as a weight of the kernel. Since eigenvalues of $K(x_i, x_j)$ must be nonnegative, we can suppose that the determinant and the trace of $H$ is also nonnegative. It is almost impossible to have eigenvalues that are all zeros.

Moreover, we can know that $H$ must be convex for the nonnegative determinant. Thus, it has the minimum.

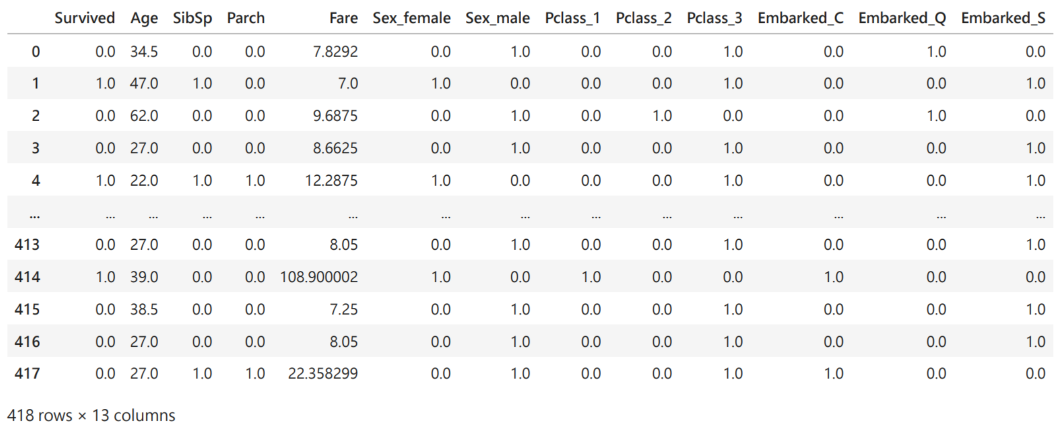
## 4 Examples / Applications

We choose two examples to elaborate on the properties of SVM.

### 4.1 Example 1: Titanic from Kaggle

#### 4.1.1 Introduction

The Titanic dataset is a popular dataset on the Kaggle. I chose it because it is complete and simple. I expect to observe the performance of SVM in a linearly separable dataset. We standardized numerical features and one-hot encoded categorical features to preprocess data. 2



| | Survived | Age | SibSp | Parch | Fare | Sex_female | Sex_male | Pclass_1 | Pclass_2 | Pclass_3 | Embarked_C | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | 34.5 | 0.0 | 0.0 | 7.8292 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| **1** | 1.0 | 47.0 | 1.0 | 0.0 | 7.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **2** | 0.0 | 62.0 | 0.0 | 0.0 | 9.6875 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **3** | 0.0 | 27.0 | 0.0 | 0.0 | 8.6625 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **4** | 1.0 | 22.0 | 1.0 | 1.0 | 12.2875 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **413** | 0.0 | 27.0 | 0.0 | 0.0 | 8.05 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **414** | 1.0 | 39.0 | 0.0 | 0.0 | 108.900002 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| **415** | 0.0 | 38.5 | 0.0 | 0.0 | 7.25 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **416** | 0.0 | 27.0 | 0.0 | 0.0 | 8.05 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **417** | 0.0 | 27.0 | 1.0 | 1.0 | 22.358299 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 |

418 rows × 13 columns

Figure 2: Preprocessed Data

### 4.1.2 Model Evaluation

Figure 3 is implemented by sklearn package with Gaussian RBF kernel, and Figure 4 is self-made with linear kernel. We also need a version with the package to certify our outcome is correct. In this simple and linear separatable dataset, we can see that the linear kernel may be slightly better than the RBF kernel by their f1 scores.

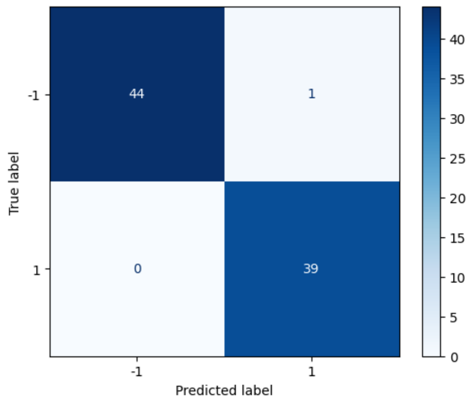


Figure 3: Confusion Matrix (RBF Kernel)

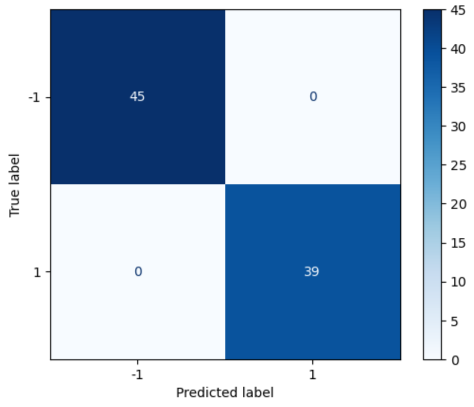


Figure 4: Confusion Matrix (Linear Kernel)

## 4.2 Example 2: Generated Moon Data

### 4.2.1 Introduction

This dataset is used to compare the Titanic with its nonlinear properties. The distribution of one class is a curve like a moon in Figure 5. SVM with the RBF kernel is also implemented by the package, while SVM with the linear kernel is self-made.

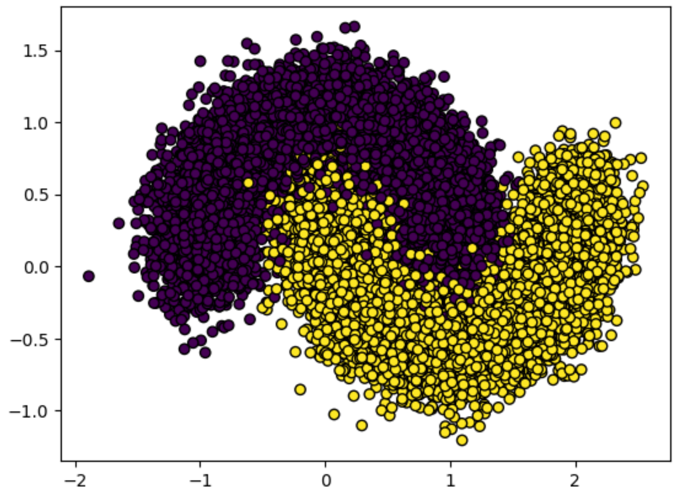


Figure 5: Generated Moon Data

### 4.2.2 Model Evaluation

In a nonlinear dataset, SVM with the RFB kernel has a great performance like Figure 6, while the accuracy of SVM with the linear kernel is shown by Figure 7, which is worse than the former one.

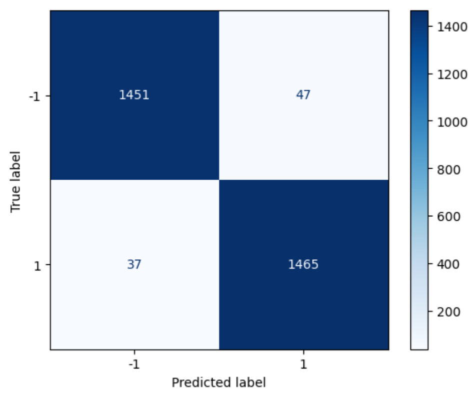


Figure 6: Confusion Matrix (RBF Kernel)

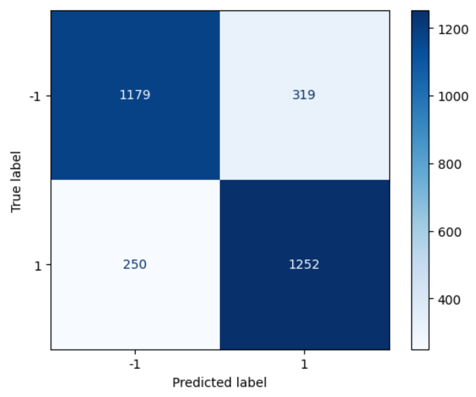


Figure 7: Confusion Matrix (Linear Kernel)

To achieve data visualization, we plot the decision boundary. From Figure 8, it can be observed that the decision boundary is a curve while it is linear in Figure 9. This difference make SVM with the RBF kernel adapt the moon data better than SVM with the linear kernel and predict more correctly.

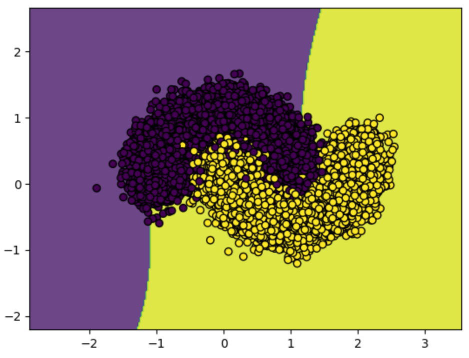


Figure 8: Confusion Matrix (RBF Kernel)

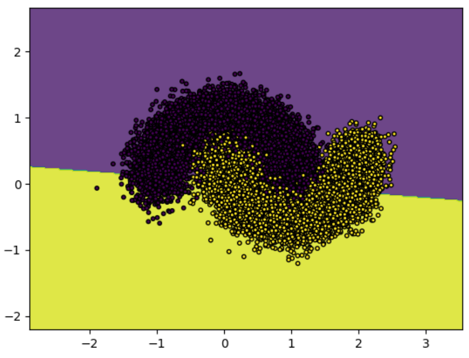


Figure 9: Confusion Matrix (Linear Kernel)

## 5 Discussions

### 5.1 Definition of SVM

From the definition of the SVM, I regard it as a traditional but powerful machine learning model. We can know how data is transformed by its kernel, and which kernel we should choose. Moreover, the feature space spanned by $x_i^T x_j$ is also a specific tool to see the correlation between data points.

We can use the feature space and the kernel to project our data on a different hyperplane. It makes SVM have decent performance on the high-dimension dataset.

### 5.2 Summary of Examples

The choice of the kernel is also crucial for optimizing predictions. It is shown that some kernels like Gaussian Radial Basis Function can turn the decision boundary from a line into a curve. However, it does not mean the RBF kernel is always better than the original linear kernel.

"All models are wrong, but some are useful." is the best quote to conclude the result. What is the space of features? Where may be a proper place for the decision boundary? How do we project it? Why? We have to always keep these questions in mind.

## References

[1] S. Amarappa and S. V. Sathyanarayana. Data classification using support vector machine (svm), a simplified approach. *International Journal of Electronics and Computer Science Engineering*, 3:435–445, 2014.

[2] M. Jain, S. Narayan, P. Balaji, A. Bhowmick, and R. K. Muthu. Speech emotion recognition using support vector machine. *arXiv preprint arXiv:2002.07590*, 2020.

[3] Vikram Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.

[4] Vojislav Kecman. Support vector machines–an introduction. In *Support Vector Machines: Theory and Applications*, pages 1–47. Springer, Berlin, Heidelberg, 2005.

[5] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, and S. Homayouni. Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:6308–6325, 2020.

[6] Kunal Talwar. On the error resistance of hinge-loss minimization. *Advances in Neural Information Processing Systems*, 33:4223–4234, 2020.

# Attachment

# Linear Algebra and Its Applications Term Project

**Hung-Ta Wang**
Institute of Industrial Engineering
National Taiwan University
r13546017@ntu.edu.tw

## 1 Rationale

### 1.1 Introduction

Machine learning has emerged as a cornerstone of modern computational science, enabling the resolution of intricate, nonlinear problems with unparalleled accuracy and adaptability across various applications.

Among these techniques, artificial neural networks (ANNs) have seen remarkable progress. Nevertheless, several inherent limitations warrant critical examination:

1. ANNs demand extensive datasets and substantial computational resources during training.
2. The complexity of ANN architectures, with numerous activation functions and layers, often compromises interpretability.
3. While achieving high predictive accuracy, ANNs may risk overfitting, questioning the true efficacy of the model.

In contrast, traditional methods like Support Vector Machines (SVMs), grounded in robust mathematical frameworks, offer superior interpretability and adaptability in specific scenarios, maintaining their relevance in the machine learning domain.

### 1.2 Motivation

The resilience of Support Vector Machines across diverse domains is well-documented. For instance, applications in speech emotion recognition [2] and remote sensing image classification [4] underscore its efficacy.

Although advanced methods such as Natural Language Processing (NLP) and Convolutional Neural Networks (CNNs) dominate modern tasks, SVM remains a reliable choice, particularly in small-sample, high-dimensional contexts. Its linear algebraic foundation ensures both robustness and interpretability, offering unique advantages in scenarios where model validation is inherently challenging [3].

Thus, this project emphasizes the theoretical and practical significance of SVMs as a foundational tool in machine learning.

## 2 Problem Background

### 2.1 Parameters

Table 1: Key Parameters Defined in This Project

| **Parameter** | **Description** |
|---|---|
| $m$ | Number of data points |
| $n$ | Number of features |
| $W_{m\times 1}$ | Weight vector perpendicular to the decision boundary |
| $w_i$ | The $i^{th}$ weight in $W_{m\times 1}$ |
| $X_{m\times n}$ | Matrix containing $m$ data points with $n$ features |
| $x_{ij}$ | Value of the $j^{th}$ feature for the $i^{th}$ data point in $X_{m\times n}$ |
| $Y_{m\times 1}$ | Vector representing the class labels of all data points |
| $y_i$ | Class label of the $i^{th}$ data point |
| $b$ | Bias term, representing the intercept of the hyperplane |
| $\xi_i$ | Distance indicating the misclassification margin for the $i^{th}$ data point |
| $C$ | Regularization parameter controlling the tolerance for classification errors |
| $l_i^{hinge}$ | Hinge loss associated with the $i^{th}$ data point |

## 2.2 Support Vector Machine

### 2.2.1 Definition

Support Vector Machines (SVMs) offer a geometrically intuitive yet mathematically rigorous framework for classification problems. Drawing upon key references such as [3] and [1], we illustrate the foundational principles of SVMs.

Figure 1, generated using Python, presents the essential concepts underpinning SVMs:

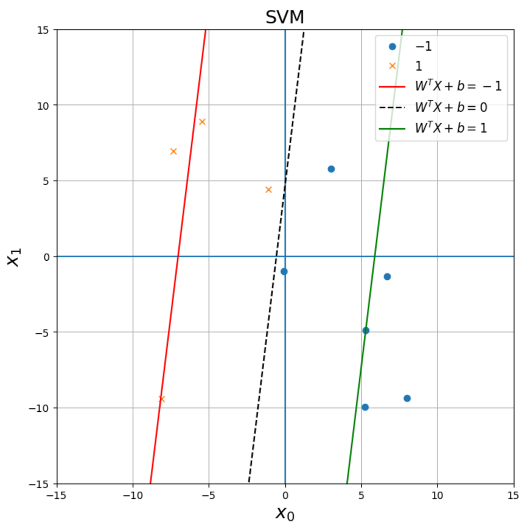


Figure 1: Illustration of SVM's Core Concepts

Three critical hyperplanes define the SVM framework:

- **Decision Boundary**: $W^Tx + b = 0$
- **Positive Class Support Plane**: $W^Tx + b = 1$
- **Negative Class Support Plane**: $W^Tx + b = -1$

The decision boundary represents the classification threshold, equidistant from the positive and negative support planes. The margin, defined as the perpendicular distance to these support planes, encapsulates the model's robustness. Equal margins on both sides are guaranteed under optimal conditions.

### 2.2.2 Primal Formulation

The margin between the decision boundary and the positive support plane is derived as:

$$
\begin{aligned}
&\frac{|(W^T X + b)_{\text{decision boundary}} - (W^T X + b)_{\text{positive support plane}}|}{||W||} \\
=&\frac{|0-1|}{||W||} = \frac{1}{||W||}.
\end{aligned}
$$

Similarly, the margin from the decision boundary to the negative support plane is:

$$
\begin{aligned}
&\frac{|(W^T X + b)_{\text{decision boundary}} - (W^T X + b)_{\text{negative support plane}}|}{||W||} \\
=&\frac{|0-(-1)|}{||W||} = \frac{1}{||W||}.
\end{aligned}
$$

The objective of SVM is to maximize this margin, expressed as:

$$
\text{maximize} \quad \frac{2}{||W||}.
$$

For computational efficiency, this is reformulated as a minimization problem:

$$
\begin{aligned}
&\text{minimize} \quad \frac{1}{2} W^T W, \\
&\text{subject to:} \\
&\quad y_i [W^T X + b] \geq 1, \quad \forall i.
\end{aligned}
$$

### 2.2.3 Dual Formulation

The primal optimization problem can be transformed into its dual form, leveraging Lagrange multipliers $\alpha_i$ to handle the constraints. The dual formulation provides additional insights and computational advantages.

The Lagrangian for the primal form is:

$$
L(W, b, \alpha) = \frac{1}{2} W^T W - \sum_{i=1}^{m} \alpha_i [y_i (W^T x_i + b) - 1], \quad \alpha_i \geq 0.
$$

To optimize $\alpha_i$, the saddle points of $W$ and $b$ are determined using the Karush-Kuhn-Tucker (KKT) conditions:

$$
\begin{aligned}
\frac{\partial L}{\partial W} = 0 &\implies W = \sum_{i=1}^{m} \alpha_i y_i x_i, \\
\frac{\partial L}{\partial b} = 0 &\implies \sum_{i=1}^{m} \alpha_i y_i = 0.
\end{aligned}
$$

Substituting $W$ back into the primal objective, the dual problem becomes:

$$\text{maximize} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j,$$

subject to:

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

#### 2.2.4 Kernel Trick for Nonlinear Problems

To extend SVM to nonlinear classification, the kernel trick is employed. This involves replacing the inner product $x_i^T x_j$ with a kernel function $K(x_i, x_j)$, which implicitly maps the input data into a higher-dimensional feature space.

Common kernel functions include:

- **Linear Kernel**: $K(x_i, x_j) = x_i^T x_j$.
- **Polynomial Kernel**: $K(x_i, x_j) = (x_i^T x_j + c)^d$.
- **Gaussian RBF Kernel**:

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

These kernels enable SVM to effectively tackle complex, nonlinear datasets.

## 3 Solutions with Linear Algebra Theories and Techniques

### 3.1 Vector Space and Eigenvalues

#### 3.1.1 Vector Space

The dataset $X_{m\times n}$ and weight vector $W_{m\times 1}$ can be represented as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}, \quad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}.$$

Key observations include:

- $\mathfrak{R}(X^T) \in \mathbb{R}^m$: The row space of $X$, spanning the data points.
- $\mathfrak{R}(X) \in \mathbb{R}^n$: The column space of $X$, spanning the features.

In scenarios where $m > n$, the rank of $X$ is constrained to $n$, suggesting potential linear dependencies among data points. This characteristic can be leveraged to reduce noise and enhance feature representations.

#### 3.1.2 Eigenvalues and Eigenvectors

For the kernel matrix $K = [K(x_i, x_j)]$, eigenvalues play a pivotal role in understanding the geometry of the transformed feature space. Positive semi-definiteness ensures:

- All eigenvalues of $K$ are nonnegative.
- The determinant and trace of $K$ provide stability metrics for the optimization process.

The eigendecomposition of $K$:

$$K = Q\Lambda Q^T,$$

where $Q$ is the matrix of eigenvectors and $\Lambda$ is the diagonal matrix of eigenvalues, facilitates efficient computations in higher-dimensional spaces.

### 3.2 Orthogonality and Projections

#### 3.2.1 Orthogonality

The term $W^T X + b$ represents the inner product $W \cdot X + b$, with its magnitude governed by $\cos\theta$. The classification behavior can be interpreted as follows:

- Large positive values indicate strong correlation with the hyperplane.
- Values near zero suggest weak discriminatory power.
- A value of exactly zero delineates the decision boundary.

#### 3.2.2 Projection Matrices

Projections onto the feature space, especially in transformed kernels, are expressed as:

$$P_\Phi = \Phi(\Phi^T \Phi)^{-1}\Phi^T,$$

where $\Phi$ represents the mapping to a higher-dimensional space. Utilizing the kernel trick, this simplifies to:

$$P_\Phi = K(K^T K)^{-1}K^T.$$

The positive semi-definiteness of $K$ guarantees computational feasibility, as the inverse of $K^T K$ exists under regularized conditions.

### 3.3 Quadratic Programming Formulation

The dual problem of SVM involves minimizing a quadratic objective function:

$$\begin{aligned}
&\text{minimize} \quad L_\alpha = -0.5\alpha^T H\alpha + f^T\alpha, \\
&\text{subject to:} \\
&\quad Y^T\alpha = 0, \quad \alpha_i \geq 0,
\end{aligned}$$

where $H = [y_i y_j K(x_i, x_j)]$ is a positive semi-definite matrix. This property ensures the convexity of the optimization landscape, allowing standard quadratic programming techniques to identify the global minimum efficiently.

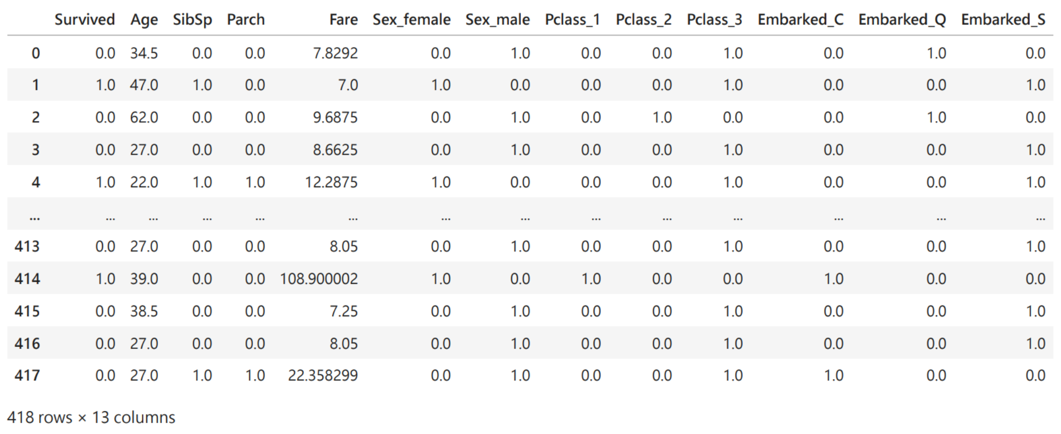
## 4 Examples / Applications

### 4.1 Example 1: Titanic Dataset

#### 4.1.1 Introduction

The Titanic dataset, renowned for its simplicity and completeness, serves as a classic benchmark. Preprocessing steps include:

- Standardizing numerical features.
- One-hot encoding categorical variables.



| | Survived | Age | SibSp | Parch | Fare | Sex_female | Sex_male | Pclass_1 | Pclass_2 | Pclass_3 | Embarked_C | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | 34.5 | 0.0 | 0.0 | 7.8292 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| **1** | 1.0 | 47.0 | 1.0 | 0.0 | 7.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **2** | 0.0 | 62.0 | 0.0 | 0.0 | 9.6875 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **3** | 0.0 | 27.0 | 0.0 | 0.0 | 8.6625 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **4** | 1.0 | 22.0 | 1.0 | 1.0 | 12.2875 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **413** | 0.0 | 27.0 | 0.0 | 0.0 | 8.05 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **414** | 1.0 | 39.0 | 0.0 | 0.0 | 108.900002 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| **415** | 0.0 | 38.5 | 0.0 | 0.0 | 7.25 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **416** | 0.0 | 27.0 | 0.0 | 0.0 | 8.05 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **417** | 0.0 | 27.0 | 1.0 | 1.0 | 22.358299 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 |

418 rows × 13 columns

Figure 2: Visualization of Preprocessed Titanic Dataset

### 4.1.2 Model Performance

Figures 3 and 4 compare the SVM model's performance using Gaussian RBF and linear kernels, respectively. Results indicate marginally superior performance of the linear kernel on this linearly separable dataset.

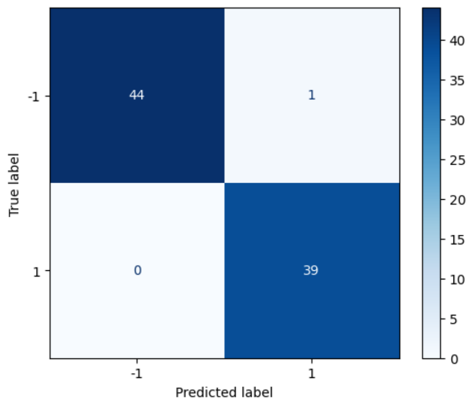


Figure 3: Confusion Matrix (RBF Kernel)

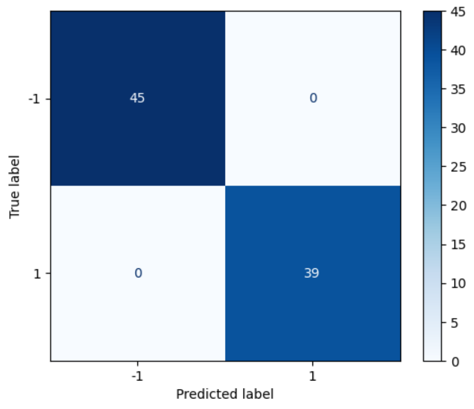


Figure 4: Confusion Matrix (Linear Kernel)

## 4.2 Example 2: Moon Dataset

### 4.2.1 Introduction

The moon dataset, characterized by its nonlinear structure, challenges the linear SVM's capabilities. The data distribution forms a crescent shape, as depicted in Figure 5.

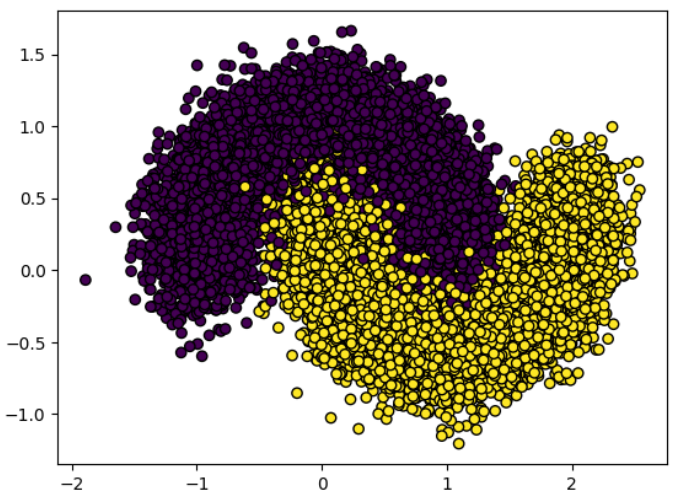


Figure 5: Visualization of Moon Dataset

### 4.2.2 Model Performance

The SVM with an RBF kernel significantly outperforms its linear counterpart, as illustrated in Figures 6 and 7.

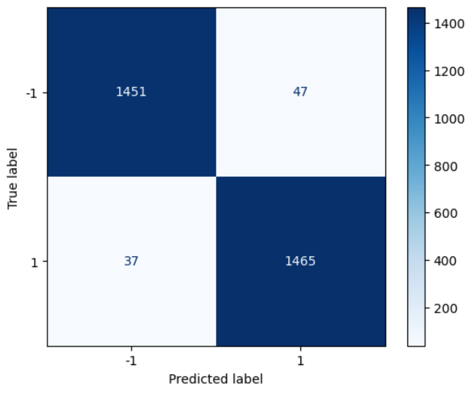


Figure 6: Confusion Matrix (RBF Kernel)

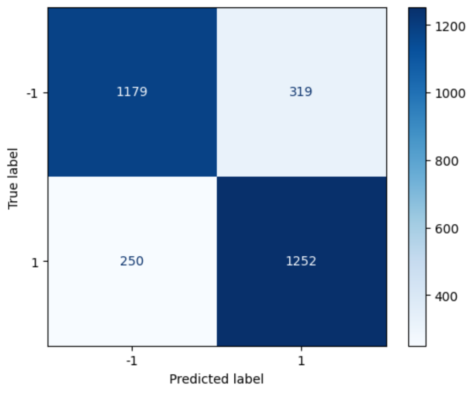


Figure 7: Confusion Matrix (Linear Kernel)

Furthermore, the decision boundary visualization highlights the advantages of the RBF kernel, as shown in Figures 8 and 9.

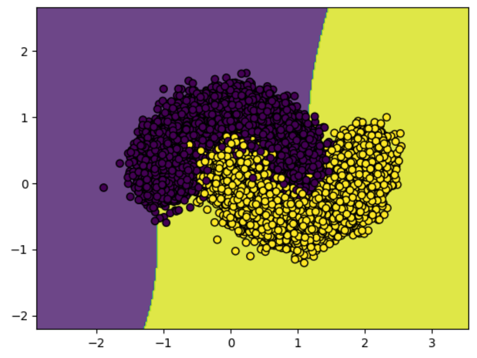


Figure 8: Decision Boundary (RBF Kernel)

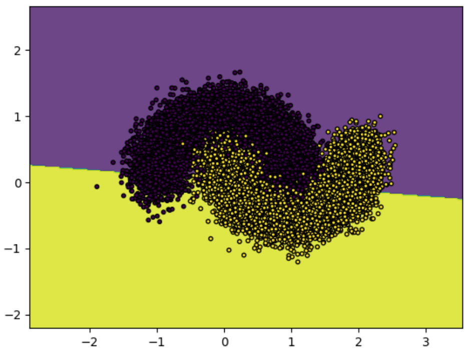


Figure 9: Decision Boundary (Linear Kernel)

## 5 Discussions

### 5.1 Interpretability of SVM

Support Vector Machines (SVMs) are rooted in a mathematically robust framework, offering unique interpretability that distinguishes them from other machine learning models, such as artificial neural networks. The following aspects highlight SVM's strengths in this regard:

- **Geometric Perspective**: SVM leverages the concept of hyperplanes to separate data points, providing an intuitive understanding of classification boundaries.
- **Kernel Transformations**: Through kernel functions, SVM maps data into higher-dimensional spaces. The interpretability of these transformations is maintained by the explicit mathematical formulation of kernels.
- **Parameter Optimization**: The optimization process, guided by the dual formulation and the Lagrange multipliers, elucidates the influence of individual support vectors on the decision boundary.

While these attributes render SVM an elegant and interpretable model, challenges such as the computational cost for large datasets and sensitivity to parameter selection should not be overlooked.

### 5.2 Insights from Examples

The empirical evaluation using the Titanic and Moon datasets offers several key takeaways:

- **Dataset Linearity**: For linearly separable data, such as the Titanic dataset, the linear kernel exhibits competitive performance, underscoring its simplicity and efficiency.
- **Nonlinear Scenarios**: The Moon dataset demonstrates the RBF kernel's adaptability to complex decision boundaries, highlighting its utility in handling nonlinear relationships.
- **Kernel Selection**: The choice of kernel significantly influences model performance. While the RBF kernel often outperforms, the linear kernel remains effective for simpler datasets, emphasizing the importance of selecting the appropriate kernel based on dataset characteristics.

These examples illustrate the practical implications of theoretical principles, bridging the gap between linear algebraic foundations and real-world applications.

### 5.3 Challenges and Future Directions

Despite its advantages, SVM faces several limitations that merit further investigation:

- **Scalability**: The quadratic complexity of solving the dual problem restricts SVM's applicability to large-scale datasets. Future research should focus on developing efficient approximation methods.
- **Parameter Sensitivity**: Hyperparameters such as the regularization parameter $C$ and kernel-specific parameters (e.g., $\sigma$ in RBF) significantly influence model performance. Automated methods for parameter tuning, such as Bayesian optimization, could enhance SVM's usability.
- **Hybrid Approaches**: Combining SVM with other machine learning techniques, such as ensemble methods or feature extraction models, may address its limitations and expand its applicability to modern challenges.

### 5.4 Conclusions

Support Vector Machines represent a cornerstone of machine learning, blending theoretical elegance with practical efficacy. This project has elucidated the fundamental principles of SVM, grounded in linear algebra, and demonstrated its applicability through real-world examples.

Key conclusions include:

- The dual formulation and kernel trick provide SVM with remarkable flexibility, enabling it to handle both linear and nonlinear classification problems effectively.
- The interpretability and mathematical rigor of SVM make it a valuable tool for researchers and practitioners, particularly in contexts where model transparency is paramount.
- Empirical evaluations underscore the importance of kernel selection and parameter tuning, highlighting the interplay between theoretical foundations and practical implementation.

While challenges such as scalability and parameter sensitivity remain, the ongoing evolution of SVM research promises to address these limitations. By bridging the gap between classical linear algebra and contemporary machine learning, SVM continues to serve as a testament to the enduring relevance of mathematical principles in solving modern problems.

## References

[1] S. Amarappa and S. V. Sathyanarayana. Data classification using support vector machine (svm), a simplified approach. *International Journal of Electronics and Computer Science Engineering*, 3:435–445, 2014.

[2] M. Jain, S. Narayan, P. Balaji, A. Bhowmick, and R. K. Muthu. Speech emotion recognition using support vector machine. *arXiv preprint arXiv:2002.07590*, 2020.

[3] Vojislav Kecman. Support vector machines–an introduction. In *Support Vector Machines: Theory and Applications*, pages 1–47. Springer, Berlin, Heidelberg, 2005.

[4] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, and S. Homayouni. Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:6308–6325, 2020.

# Linear Algebra Final Report Code Implementation

## Hung-Ta, Wang | Student ID: R13546017 | HW_ID: 56

In [52]:
```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.datasets import make_moons
```

### Example 1: Titanic from Kaggle

#### Example 1: Data Preprocessing

In [53]:
```
boat = pd.read_csv("titanic.csv")
print("Step 1: Clear categorical features with too many different values")
print("")
Ticket_list = []
for i in boat["Ticket"]:
    if i not in Ticket_list:
        Ticket_list.append(i)
print("Types of Ticket: ", len(Ticket_list))

Name_list = []
for i in boat["Name"]:
    if i not in Name_list:
        Name_list.append(i)
print("Types of Name: ", len(Name_list))

PassengerId_list = []
for i in boat["PassengerId"]:
    if i not in PassengerId_list:
        PassengerId_list.append(i)
print("Types of PassengerId: ", len(PassengerId_list))

columns_to_drop = ["Ticket", "Name", "PassengerId"]

print("")
print("If we encode Ticket, Name, PassengerId, the matrix would be too sparse to analyze.")
print("To prevent from being costly or overfitting, we delete them.")
print("-------------------------------------------------------------------------------------------------")
print("Step 2: Clear features with too many NaN values")
print("")

Cabin_counter = 0
for i in boat["Cabin"].isna():
    if i:
        Cabin_counter += 1
print("The number of NaN values in Age: ", Cabin_counter)

Age_counter = 0
for i in boat["Age"].isna():
    if i:
        Age_counter += 1
print("The number of NaN values in Age: ", Age_counter)

columns_to_drop = ["Ticket", "Name", "PassengerId", "Cabin"]

print("")
print("Since the data in Cabin is too sparse to provide useful information, we delete it.")
print("To complete Age and Fare, We use a simple method: fill the median in.")
boat["Age"] = boat["Age"].fillna(boat["Age"].median())
boat["Fare"] = boat["Fare"].fillna(boat["Fare"].median())

boat.drop(columns=columns_to_drop, inplace=True)

print("-------------------------------------------------------------------------------------------------")

print("Step 3: Transform categorical data into one-hot encoding form")
print("")
print("We transform Sex, Pclass and Embarked into one-hot encoding form, and turn it into float32")
boat = pd.get_dummies(boat, columns=["Sex", "Pclass", "Embarked"], prefix=["Sex", "Pclass", "Embarked"])
boat = boat.astype("Float32")
print("-------------------------------------------------------------------------------------------------")
print("Final result is shown below: ")
print("")
boat

# boat.to_csv("LA.csv", index=None)
```

```
Step 1: Clear categorical features with too many different values

Types of Ticket:  363
Types of Name:  418
Types of PassengerId:  418

If we encode Ticket, Name, PassengerId, the matrix would be too sparse to analyze.
To prevent from being costly or overfitting, we delete them.
-------------------------------------------------------------------------------------------------
Step 2: Clear features with too many NaN values

The number of NaN values in Age:  327
The number of NaN values in Age:  86

Since the data in Cabin is too sparse to provide useful information, we delete it.
To complete Age and Fare, We use a simple method: fill the median in.
-------------------------------------------------------------------------------------------------
Step 3: Transform categorical data into one-hot encoding form

We transform Sex, Pclass and Embarked into one-hot encoding form, and turn it into float32
-------------------------------------------------------------------------------------------------
Final result is shown below:
```

Out[53]:

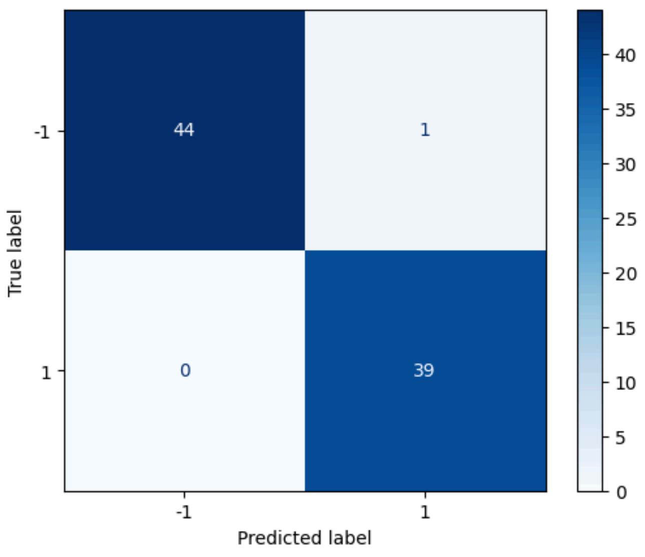
| | Survived | Age | SibSp | Parch | Fare | Sex_female | Sex_male | Pclass_1 | Pclass_2 | Pclass_3 | Embarked_C | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 34.5 | 0.0 | 0.0 | 7.8292 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| 1 | 1.0 | 47.0 | 1.0 | 0.0 | 7.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| 2 | 0.0 | 62.0 | 0.0 | 0.0 | 9.6875 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| 3 | 0.0 | 27.0 | 0.0 | 0.0 | 8.6625 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| 4 | 1.0 | 22.0 | 1.0 | 1.0 | 12.2875 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 413 | 0.0 | 27.0 | 0.0 | 0.0 | 8.05 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| 414 | 1.0 | 39.0 | 0.0 | 0.0 | 108.900002 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 415 | 0.0 | 38.5 | 0.0 | 0.0 | 7.25 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| 416 | 0.0 | 27.0 | 0.0 | 0.0 | 8.05 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| 417 | 0.0 | 27.0 | 1.0 | 1.0 | 22.358299 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 |

418 rows × 13 columns

Example 1: Use sklearn SVC packge

In [54]:
```python
def make_cm(cm):
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[-1, 1])
    disp.plot(cmap=plt.cm.Blues)
    plt.show()
```

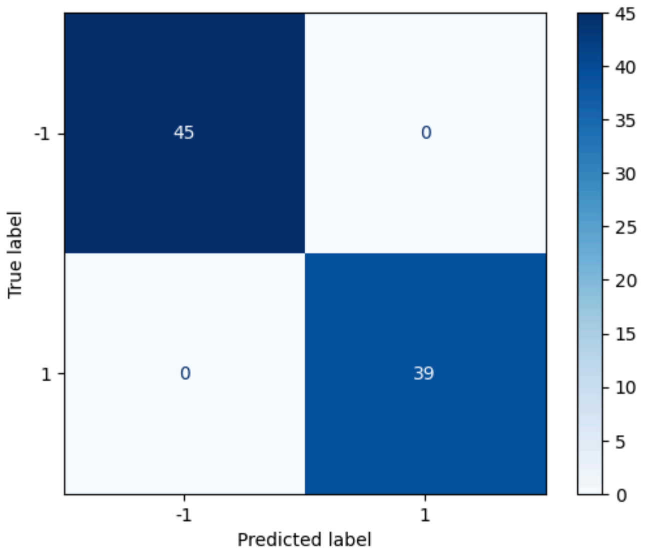
In [55]:
```python
x_numerical = boat.iloc[:, 1:5].values
x_categorical = boat.iloc[:, 5:].values
y = boat.iloc[:, 0].values
y = np.where(y == 1, 1, -1)


scaler = StandardScaler()
x_numerical_transformed = scaler.fit_transform(x_numerical)
x = np.hstack((x_numerical_transformed, x_categorical))

x = x.astype(float)
y = y.astype(float)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

svm_model = SVC(kernel='rbf', C=1.0, random_state=0)
svm_model.fit(x_train, y_train)

y_pred = svm_model.predict(x_test)

cm = confusion_matrix(y_test, y_pred)

make_cm(cm)
```

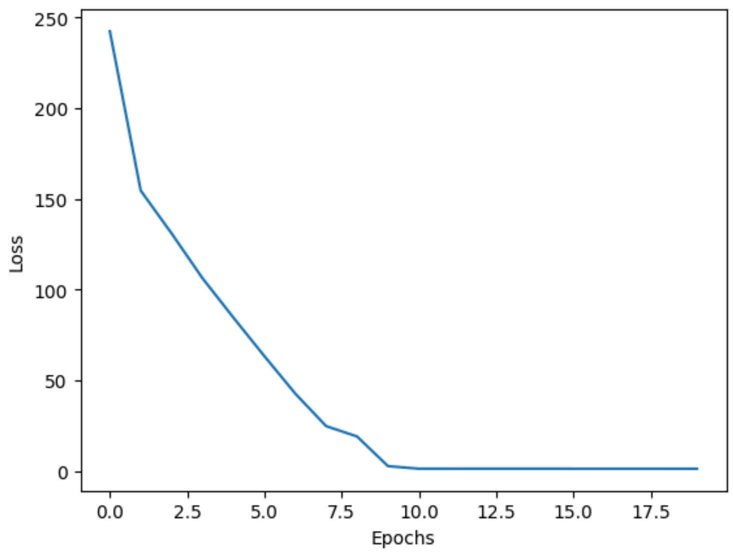


Example 1: SVM Implementation

In [56]:
```python
def hinge_loss(x, y, w, b, C):
    N = len(y)
    loss = 0.5 * np.dot(w, w) + C * np.sum(np.maximum(0, 1 - y * (np.dot(x, w) + b)))
    return loss

def compute_gradient(x, y, w, b, C):
    N = len(y)
    grad_w = w.copy()
    grad_b = 0
    for i in range(N):
        margin = y[i] * (np.dot(x[i], w) + b)
        if margin < 1:
            grad_w -= C * y[i] * x[i]
            grad_b -= C * y[i]
    return grad_w, grad_b

def train_svm(x, y, C=1.0, learning_rate=0.01, epochs=1000):
    n_features = x.shape[1]
    w = np.zeros(n_features)
    b = 0
    losses = []
    for epoch in range(epochs):
        grad_w, grad_b = compute_gradient(x, y, w, b, C)
        w -= learning_rate * grad_w
        b -= learning_rate * grad_b
        loss = hinge_loss(x, y, w, b, C)
        losses.append(loss)
    return w, b, losses

def predict(x, w, b):
    return np.sign(np.dot(x, w) + b)

def plot_loss(epochs, losses):
    plt.plot(range(epochs), losses)
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.show()
```

In [57]:
```python
x_numerical = boat.iloc[:, 1:5].values
x_categorical = boat.iloc[:, 5:].values
y = boat.iloc[:, 0].values
y = np.where(y == 1, 1, -1)

scaler = StandardScaler()
x_numerical_transformed = scaler.fit_transform(x_numerical)
x = np.hstack((x_numerical_transformed, x_categorical))

x = x.astype(float)
y = y.astype(float)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

C = 1.0
learning_rate = 0.001
epochs = 20
w, b, losses = train_svm(x_train, y_train, C, learning_rate, epochs)

y_pred = predict(x_test, w, b)

cm = confusion_matrix(y_test, y_pred)

make_cm(cm)
plot_loss(epochs, losses)
```





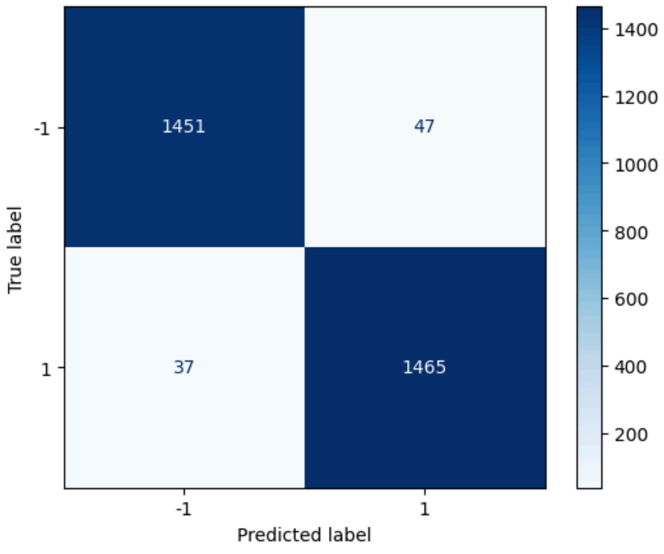
## Example 2: Generated Moon Data

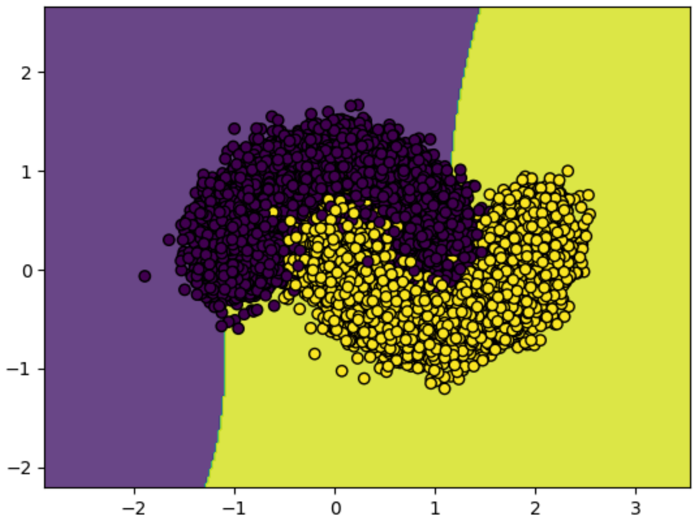
Example 2: Use sklearn SVC packge

In [58]:
```python
def plot_decision_boundary_model(x, y, model):
    x_min, x_max = x[:, 0].min() - 1, x[:, 0].max() + 1
    y_min, y_max = x[:, 1].min() - 1, x[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300), np.linspace(y_min, y_max, 300))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.8, cmap="viridis")
    plt.scatter(x[:, 0], x[:, 1], c=y, edgecolor="k", cmap="viridis")
    plt.show()

def plot_decision_boundary_svm(x, y, w, b):
    x_min, x_max = x[:, 0].min() - 1, x[:, 0].max() + 1
    y_min, y_max = x[:, 1].min() - 1, x[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300), np.linspace(y_min, y_max, 300))
    Z = np.sign(np.dot(np.c_[xx.ravel(), yy.ravel()], w) + b).reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.8, cmap="viridis")
    plt.scatter(x[:, 0], x[:, 1], c=y, edgecolor="k", cmap="viridis", s=10)
    plt.show()
```

In [61]:
```python
x, y = make_moons(n_samples=20000, noise=0.2, random_state=0)
y = np.where(y == 1, 1, -1)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state=0)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

svm_model = SVC(kernel='rbf', C=1.0, random_state=0)
svm_model.fit(x_train, y_train)

y_pred = svm_model.predict(x_test)

cm = confusion_matrix(y_test, y_pred)
make_cm(cm)
```

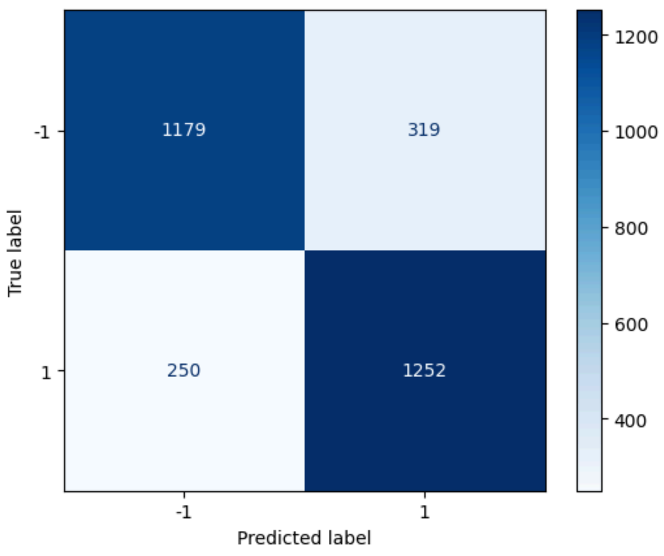
```
plot_decision_boundary_model(x, y, svm_model)
```



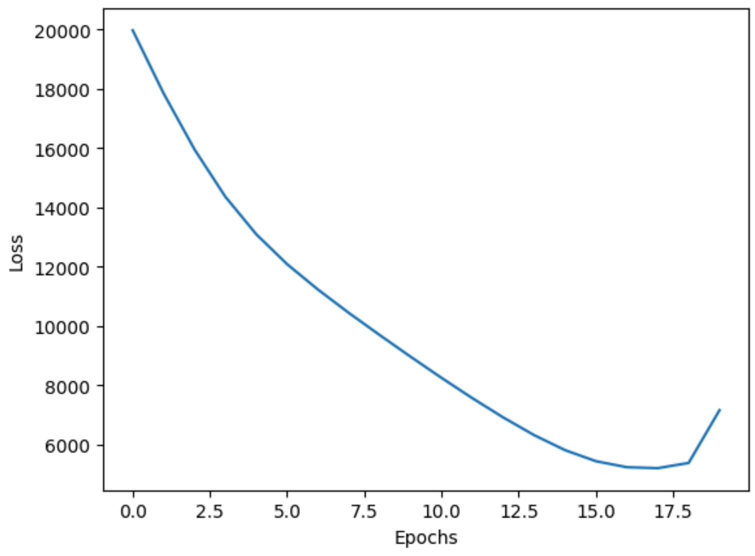


Example 2: SVM Implementation

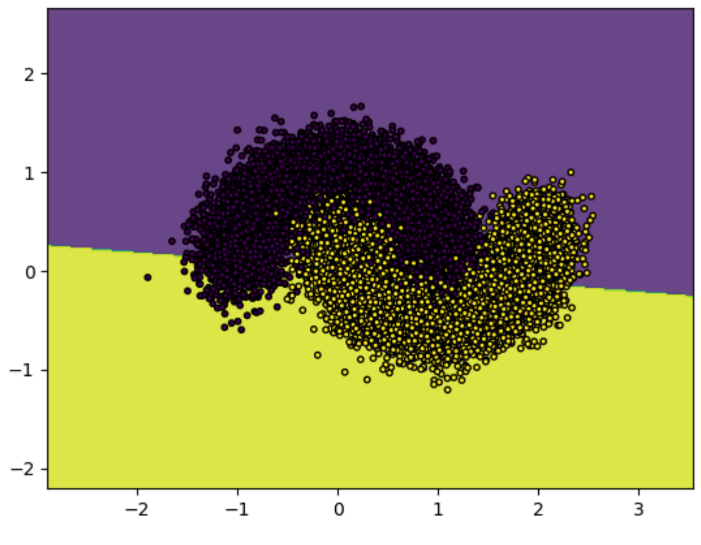
```
In [60]: x, y = make_moons(n_samples=20000, noise=0.2, random_state=0)
y = np.where(y == 1, 1, -1)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state=0)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

C = 1.0
learning_rate = 0.001
epochs = 20
w, b, losses = train_svm(x_train, y_train, C, learning_rate, epochs)

y_pred = predict(x_test, w, b)
cm = confusion_matrix(y_test, y_pred)

make_cm(cm)
plot_loss(epochs, losses)

plot_decision_boundary_svm(x, y, w, b)
```



1179
319
250
1252
-1
1
True label
Predicted label
1200
1000
800
600
400



Loss
Epochs
20000
18000
16000
14000
12000
10000
8000
6000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



2
1
0
−1
−2
−2
−1
0
1
2
3