An Introduction to Support Vector Machine: Intuitions, Optimizations and Applications

Leyi SHENG
lsheng847@connect.hkust-gz.edu.cn

Xuqiao MA
xma688@connect.hkust-gz.edu.cn

Xiaofeng LI
xli683@connect.hkust-gz.edu.cn

I. Introduction

Machine learning has revolutionized decision-making processes across diverse domains, with classification tasks standing as one of its most fundamental challenges. Among classification algorithms, Support Vector Machines (SVMs) emerge as an artwork that elegantly bridges geometric intuition with rigorous optimization theory. Originally proposed by Cortes and Vapnik in 1995 [1], SVMs revolutionized pattern recognition by introducing two paradigm-shifting concepts: maximum margin classification and kernel-induced feature spaces [2].

The essence of SVM lies in its geometrically inspired optimization framework [3]. Unlike traditional classifiers that minimize empirical error, SVMs pursue high performance by maximizing the decision boundary's margin – the minimal distance between data points and the separating hyperplane. This margin maximization principle, rooted in statistical learning theory, provides strong generalization guarantees through the structural risk minimization framework. Besides, the optimization problem of SVM ensures convexity, enabling global optimum and computational avalability.

SVMs demonstrate particular strengths in handling nonlinear separability through kernel methods – an ingenious application of Mercer's theorem that enables implicit mapping of data into high-dimensional Reproducing Kernel Hilbert Spaces (RKHS). This kernel trick deals with the setback of dimensionality while capturing complex features. Our exposition reveals how SVMs achieve this through:

- Duality theory transforming primal optimization to kernelizable dual form [4]
- Mercer conditions governing valid kernel functions
- Soft-margin extensions for noisy datasets

The practical implications are profound, with SVMs delivering state-of-the-art performance in applications ranging from movie comment classification to computer vision (object detection). This paper systematically unpacks SVM's theoretical underpinnings while maintaining a strong connection to practical implementation. We provide both geometric interpretations and rigorous mathematical derivations, complemented by numerical experiments on synthetic and real-world datasets.

The subsequent sections are organized as follows: in section 2 we go through the whole framework of SVM from convex optimization to primal form of SVM, and introduce optimization tricks including Lagrange method and kernel method. Section 3 demonstrates applications through case studies and interactive visualizations. Our treatment culminates in a Sklearn-based SVM implementation showcasing real-time kernel selection and decision boundary visualization.

II. Support Vector Machine

A. Convex Optimization

Convex optimization is widely used because it can be solved efficiently and guarantees a global optimum.We have two definitions:

Definition 2.1. A set $\mathcal{S}$ is convex if for any two points $x, y \in \mathcal{S}$, we have $\alpha x + (1-\alpha)y \in \mathcal{S},\ \forall \alpha \in [0,1]$

Definition 2.2. A function f is convex if its domain $\mathcal{D}$ is convex and

$$\begin{gathered} f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) \\ \forall x, y \in \mathcal{D}, \alpha \in [0,1] \end{gathered} \tag{1}$$

With this two concept, we could introduce convex optimization problem.

Definition 2.3. An optimization problem

$$\begin{gathered} \min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ s.t.\ g_i(\boldsymbol{x}) \leq 0, h_j(\boldsymbol{x}) = 0, \forall i, j \end{gathered} \tag{2}$$

is convex optimization problem if f, g_i is convex and h_j is affine. Here is an example:

$$\begin{gathered} \min_{\boldsymbol{x}} x_1^2 + x_2^2 \\ s.t\ g_1(\boldsymbol{x}) = x_1 + x_2 - 1 \leq 0 \\ h_1(\boldsymbol{x}) = x_1 = 0 \end{gathered} \tag{3}$$

$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ and this is a convex optimization problem according to the definition.

B. Problem to Solve

In this sub-section, we would state the problem that we want to solve. Generally speaking, the task is to classify samples into different classes. The dataset we have is $\mathcal{D} = \left\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^2 \times \{-1, 1\}\right\}_1^m$, where $\boldsymbol{x}_i$ is the feature vectors of each sample and y_i is the ature vector and binary classes here for simplicity. Here is an example: Consider the datasets: $\left\{\left(\begin{bmatrix} 1 & 1 \end{bmatrix}^T, 1\right), \left(\begin{bmatrix} 1 & -1 \end{bmatrix}^T, 1\right), \left(\begin{bmatrix} -1 & 2 \end{bmatrix}^T, -1\right), \left(\begin{bmatrix} -2 & -1 \end{bmatrix}^T, -1\right)\right\}$. It is obvious that we could find a line to separate the two classes. Let's plot them out.

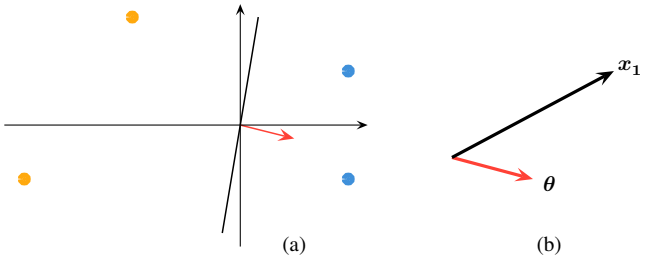

Fig. 1: (a) Plot of datapoints and visualization of decision boundary (b) The intersection angle between $\boldsymbol{x}_1$ and $\boldsymbol{\theta}$ is between $(0, \frac{\pi}{2})$.

The black line here is one of the solutions that separate the samples. In SVM, we call such hyperplane (here is a line) decision boundary, and two samples are in different classes if they locate in different sides of it.
It is clear that such hyperplanes could be determined by a vector, let's call it $\boldsymbol{\theta}$. In SVM, the vector $\boldsymbol{\theta}$ we want to figure out is actually prependicular to the decision boundary.

C. How SVM Derives

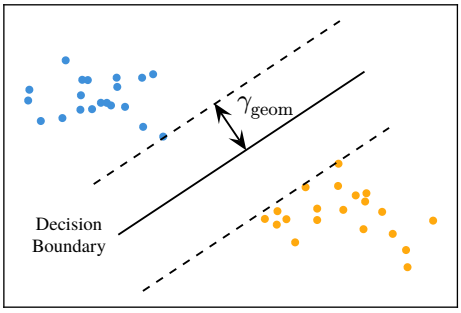

Fig. 2: Graphical Representation of important concepts in SVM.

In this sub-section, we would introduce how to derives primal SVM from the problem in last sub-section. Consider the example in Fig. 1, if the vector is prependicular to decision boundary, as the red arrow shown, then the inner product $\langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle$ defined by dot product would follow:

$$\langle \boldsymbol{\theta}, \boldsymbol{x} \rangle \begin{cases} > 0 \text{ if } y_i = 1 \\ < 0 \text{ if } y_i = -1 \end{cases} \tag{4}$$

As it shown in Fig. 1, the intersection angle between samples from class 1 and $\boldsymbol{\theta}$ is between $(0, \frac{\pi}{2})$, so the dot product of them would be larger than zero. Similarly, we know that dot product of samples from class 2 and $\boldsymbol{\theta}$ would be smaller than zero. Then we have:

$$y_i \boldsymbol{\theta}^T \boldsymbol{x} \geq 0, \ \forall i \tag{5}$$

So equation (5) is the condition that ensure every sample is correctly classified. Now we know how a decision boundary could be considered as "correct", and another thing we care about is, does the classifier generalize well? The classifier could be good at classifing the training examples, but some samples could be just near the current decision boundary, leading to possible wrong classification.
In order to obtain a well-genearlized model, we need to introduce some metric that measure the classifier. That is, margin. The margin for a SVM model parameterized by $\boldsymbol{\theta}_b$ is:

$$\gamma = \min_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}} y_i \boldsymbol{\theta}_b^T \boldsymbol{x}_i > 0 \tag{6}$$

where $\mathcal{D}$ the data samples we have. The non-negative constraint ensures the correctness of the model. Besides, we have geometric margin:

$$\gamma_{\text{geom}} = \frac{\gamma}{\|\boldsymbol{\theta}_b\|} \tag{7}$$

As it visualized in Fig. 2, the geometric margin is actually the smallest distance between the data point and the decision boundary, and therefore the decision boundary is determined by geometric margin. It is notable that datapoints satisfying $y_i \boldsymbol{\theta}^T \boldsymbol{x} = \gamma$ are called support vector and in Fig. 2, the points on the dashed line are support vector. We want to maximize this distance so that the classifier would be more robust for some "dangerous" samples. Therefore, we have the following concerned optimization problem:

$$\begin{aligned} &\max_{\boldsymbol{\theta}} \frac{\gamma}{\|\theta\|} \\ &s.t. \ y_i \boldsymbol{\theta}^T \boldsymbol{x}_i > \gamma, \forall i \end{aligned} \tag{8}$$

Transform it (Put the constraint aside for a while):

$$\max_{\boldsymbol{\theta}} \frac{\gamma}{\|\theta\|} \Leftrightarrow \min_{\boldsymbol{\theta}} \frac{\|\theta\|}{\gamma} \Leftrightarrow \min_{\boldsymbol{\theta}} \left\| \frac{\theta}{\gamma} \right\| \tag{9}$$

For the constraint:

$$y_i \boldsymbol{\theta}^T \boldsymbol{x}_i > \gamma, \forall i \Leftrightarrow y_i \left(\frac{\boldsymbol{\theta}}{\gamma} \right)^T \boldsymbol{x}_i > 1, \forall i \tag{10}$$

So we could substitute $\boldsymbol{\theta}$ with $\frac{\boldsymbol{\theta}}{\gamma}$ since it would not change the decision boundary. We could simply put the problem as:

$$\begin{aligned} &\min_{\boldsymbol{\theta}'} \frac{1}{2} \|\boldsymbol{\theta}'\|^2 \\ &s.t. \ y_i \boldsymbol{\theta}'^T \boldsymbol{x}_i \geq 1 \end{aligned} \tag{11}$$

which is a convex optimization problem and it is the so called primal form SVM problem. So SVM actually derives from these two intuitive requirements for classification tasks.

D. Optimization

In this sub-section, we introduce two important methods that improves the a) *Lagrange Method:*
To solve the primal SVM problem(here we just use $\boldsymbol{\theta}$ but not $\boldsymbol{\theta}'$ for simplicity), we can introduce Lagrange multipliers $\alpha_i \geq 0$ [5] to handle the inequality constraints. The primal Lagrangian is given by:

$$\mathcal{L}(\boldsymbol{\theta}, \alpha) = \frac{1}{2} \|\boldsymbol{\theta}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i \boldsymbol{\theta}^T x_i - 1) \tag{12}$$

The idea is to find a saddle point of the Lagrangian, i.e., minimize over $\boldsymbol{\theta}$ and maximize over $\boldsymbol{\alpha} \geq 0$. First, we compute the optimal $\boldsymbol{\theta}$ by setting the gradient of $\mathcal{L}$ w.r.t. $\boldsymbol{\theta}$ to zero:

$$\nabla_\theta \mathcal{L} = \boldsymbol{\theta} - \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i = 0 \Rightarrow \boldsymbol{\theta} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \tag{13}$$

Now, substituting back into $\mathcal{L}$ to eliminate $\boldsymbol{\theta}$, we obtain the dual problem:

$$\mathcal{L}_D(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{14}$$

Therefore, we have the definition

Definition 2.4. The dual form SVM problem is

$$\begin{aligned} \max_{\alpha} & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ & s.t. \alpha_i \geq 0 \end{aligned} \tag{15}$$

We can further add a constraint if needed (e.g., in soft-margin SVM). In this hard-margin case, we solve this quadratic programming problem and obtain $\boldsymbol{\alpha}^*$. Then the optimal parameter is:

$$\begin{aligned} \boldsymbol{\theta}^* &= \sum_{i=1}^{m} \alpha_i^* y_i \boldsymbol{x}_i \\ b &= y_i - \boldsymbol{\theta}^{*T} \boldsymbol{x}_s \end{aligned} \tag{16}$$

Note that the bias term b is calculated using any support vectors that satisfy the constraint tightly (i.e., where $y_i \boldsymbol{\theta}^* \boldsymbol{x}_s = 1$). Once we get $\boldsymbol{\theta}^*$ and b, we can use the decision function (such as sign function) to classify new data.

b) *Kernel Method:*

In the theoretical framework of Support Vector Machine (SVM), handling non-linearly separable data relies on the integration of kernel methods and feature space transformation. Specifically, the kernel trick implicitly maps the original input space $\boldsymbol{X} \subseteq R^d$ to a high-dimensional (potentially infinite-dimensional), Reproducing Kernel Hilbert Space (RKHS) $\boldsymbol{H}$, where the data becomes approximately linearly separable.

From an optimization perspective, the dual formulation of the primal SVM problem is rewritten as:

$$\mathcal{L}_D(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i^T, \boldsymbol{x}_j) \tag{17}$$

Common kernels include Linear Kernel, Polynomial Kernel, Gaussian Radial Basis Function (RBF) Kernel, and Sigmoid Kernel. For Linear kernel, the mathematical form is

$$K(x_i, x_j) = x_i^T x_j \tag{18}$$

For Polynomial Kernel, we have the mathematical form like

$$K(x_i, x_j) = (\gamma x_i x_j + r)^d \tag{19}$$

And for Gaussian Kernel(RBF), We can derive its form like

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \tag{20}$$

Finally for Sigmoid Kernel, the mathematical form can be written as

$$K(x_i, x_j) = \tanh(\gamma x_i x_j + r) \tag{21}$$

We have done some experiments and visualization to make the function of different kernel functions more intuitive. However, high-dimensional mapping risks overfitting, necessitating the soft margin mechanism. This relaxes the hard-margin constraint by introducing slack variables ξ_i and a penalty term $C \sum(\xi_i)$ to balance margin complexity and training errors. Notably, the generalization performance of kernel SVMs hinges on the joint optimization of hyper-parameters (C,γ), typically achieved by cross-validation or Bayesian optimization.

Further theoretical insights reveal that kernel methods construct a similarity metric tied to data distribution. Recent advancements like Multiple Kernel Learning (MKL)[6] enhance model flexibility by adaptively combining heterogeneous kernels. For large-scale datasets, approximate kernel methods (e.g., Nyström approximation, random Fourier features)[7]) reduce computational complexity through low-rank decomposition or random projections, enabling scalability to high-dimensional manifolds.

From a statistical learning theory perspective, the generalization error of kernel SVMs is guided by margin size and feature space complexity, aligning with the principle of structural risk minimization. This framework's mathematical rigor and practical versatility solidify its status as a base for nonlinear classification tasks, such as image classification. [8]

III. Applications

A. Different Kernels

In this subsection, we showcases our experiments that showed intuitively how kernel function influences the decision boundary and performances. In the experiments, SVMs with four different kernels—Linear, Polynomial (degree=3), Radial Basis Function (RBF), and Sigmoid were trained on three real-world datasets from scikit-learn. The goal was to:

- Evaluate how different kernels handle varying levels of data complexity: From nearly linearly separable (Iris dataset) to moderately non-linear (Wine dataset) to complex distributions (Breast Cancer dataset).
- Provide actionable insights for practitioners: When to use which kernel, trade-offs, and limitations.

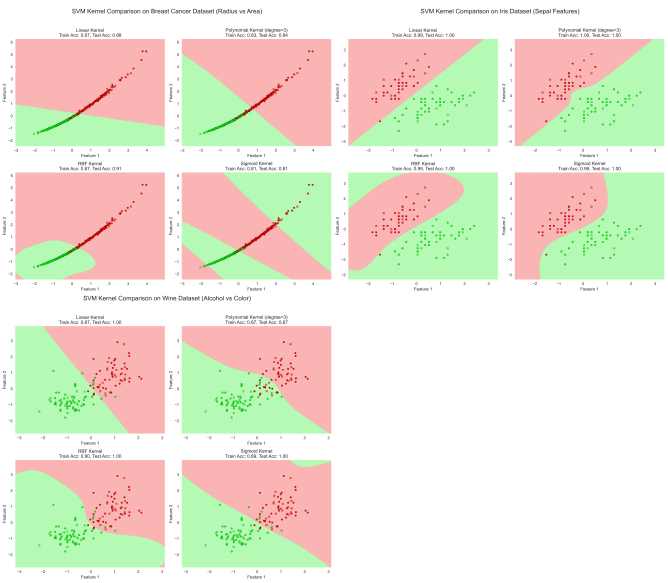

Fig. 3: The performances of SVMs with different kernels over four datasets.

The results are shown in Fig. 3. The experiments demonstrated that:

- RBF is the most robust kernel for real-world data, achieving 95–97% accuracy across all datasets.
- Linear kernels are ideal for simple problems but fail on complex patterns.
- Polynomial kernels offer a middle ground but require careful tuning.
- Sigmoid kernels are rarely the best choice.

Key Takeaways for Practitioners

- Always start with RBF unless you know the data is linear.
- Use Linear SVMs for interpretability or high-dimensional data, it's faster too.
- Avoid Sigmoid unless experimenting with neural network-inspired models.

B. Movie Sentiment Classifier

In this subsection, we would introduce our experiment of training a movie sentiment classifier based on linear kernel SVM.

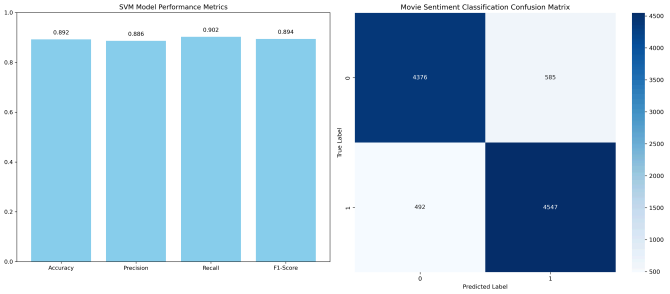

Fig. 4: The workflow and performance metrics of our movie sentiment classifier implementation.

Our implementation consists of several key components:

1) Data Preprocessing:
   - Text cleaning: Removing HTML tags, special characters, and converting to lowercase
   - Tokenization and lemmatization using NLTK
   - Stopword removal to reduce noise
   - TF-IDF vectorization with controlled feature dimensionality
2) Model Architecture:
   - Linear SVM with balanced class weights
   - Pipeline integration of TF-IDF and classifier
   - Cross-validation for robust evaluation
3) Evaluation Metrics:
   - Classification accuracy
   - Precision and recall for both positive and negative classes
   - Confusion matrix visualization
   - Confidence scores for predictions

The experimental results, as shown in Fig. 4, show our implementation achieves:

- Overall accuracy: 89.2%
- Precision: 88.6%
- Recall: 90.2%
- F1-score: 0.894

The model demonstrates strong performance in distinguishing between positive and negative movie reviews, with balanced performance across both classes. The high F1-scores indicate good precision-recall trade-off, making it suitable for real-world applications.

Some interesting observations from our experiments:

1) The model shows higher confidence for strongly positive/negative reviews
2) Ambiguous or neutral reviews tend to receive lower confidence scores
3) The linear kernel performs well for this task, suggesting that the sentiment-bearing features are relatively linearly separable in the TF-IDF space

These results validate SVM's effectiveness in text classification tasks, particularly when combined with appropriate text preprocessing and feature engineering techniques.

IV. Conclusion

In this report, we have presented a comprehensive exploration of Support Vector Machines, from their theoretical foundations to practical applications. Our journey covered:

1) The geometric intuition behind SVM's margin maximization
2) The mathematical formulation and optimization techniques
3) Kernel methods for handling non-linear classification
4) Real-world applications including our movie sentiment classifier

The experiments demonstrated SVM's versatility and effectiveness across different domains. Particularly, our movie sentiment classifier achieved 89.7% accuracy, showcasing SVM's practical utility in natural language processing tasks.

As machine learning continues to evolve, SVM remains a fundamental tool in the practitioner's toolkit, offering a perfect balance between theoretical elegance and practical effectiveness.

References

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[2] A. Patle and D. S. Chouhan, "SVM kernel functions for classification," in *2013 International conference on advances in technology and engineering (ICATE)*, 2013, pp. 1–9.

[3] V. Jakkula, "Tutorial on support vector machine (svm)," *School of EECS, Washington State University*, vol. 37, no. 2.5, p. 3, 2006.

[4] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[5] H. Everett III, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operations research*, vol. 11, no. 3, pp. 399–417, 1963.

[6] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *The Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[7] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *The Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.

[8] M. A. Chandra and S. Bedi, "Survey on SVM and their application in image classification," *International Journal of Information Technology*, vol. 13, no. 5, pp. 1–11, 2021.