Course EE5904 Neural Networks Part II

Project 1: SVM for Classification of Spam Email Messages

Lecturer: **Assoc. Prof. Peter C. Y. Chen**

Author: XU YIMIAN
Student ID: A0295779Y
Email: e1349917@u.nus.edu

National University of Singapore

Apr 6, 2025

Contents

List of Tables

List of Figures

Task 1

1. Discriminant Function

$$g(\mathbf{x}) = \mathbf{w}_\circ^T \mathbf{x} + b_\circ$$

where $\mathbf{w} \in \Re^d$ is normal vector of the decision hyperplane, $b \in \Re$ is the bias (offset from origin). The decision rule is: prediction $\hat{y} = \text{sign}(g(x))$.

In SVM, the discriminant function is a mathematical expression used to assign a label to a given input sample. It computes a score whose sign determines the class of the input: a positive value implies classification into one class (e.g., +1), a negative value implies classification into the other class (e.g., -1), the magnitude of the value indicates the confidence (or distance from the decision boundary). In other word, the function represents the signed distance from the sample to the separating hyperplane.

We have the dual formulation with transformation $\varphi(\mathbf{x})$ as follows:

$$g(\mathbf{x}) = \mathbf{w}_\circ^T \varphi(\mathbf{x}) + b_\circ$$

also can be rewritten as:

$$g(\mathbf{x}) = \sum_{i=1}^{N} \alpha_{\circ,i} d_i \varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}) + b_\circ = \sum_{i=1}^{N} \alpha_{\circ,i} d_i K(\mathbf{x}_i, \mathbf{x}) + b_\circ$$

2. Task Requirements

The discriminant function is formulated as:

$$g(\mathbf{x}) = \sum_{i=1}^{N} \alpha_{\circ,i} d_i K(\mathbf{x}_i, \mathbf{x}) + b_\circ$$

(a) Hard-margin SVM with **linear** kernel

$$g(\mathbf{x}) = \sum_{i=1}^{N} \alpha_{\circ,i} d_i \mathbf{x}_i^T \mathbf{x} + b_\circ$$

(b) Hard-margin SVM with polynomial kernel

$$g(\mathbf{x}) = \sum_{i=1}^{N} \alpha_{\circ,i} d_i (\mathbf{x}_i^T \mathbf{x} + 1)^p + b_\circ$$

(c) Soft-margin SVM with polynomial kernel

The formula is the same as (b).

To find the optimal values $\alpha_{\circ,i}$ and $b_\circ$, we need to solve the convex quadratic programming (QP) problem mentioned above. I implement the training programming in mfile `train_svm.m`, and define the discriminant function $g(\mathbf{x})$ in mfile `predict.m`, where it is used to predict the label of a sample. Compute the kernel (both linear and nonlinear) by a function in mfile `kernel.m`. For more details about the SVM prime optimization problem and dual problem, please refer to Appendix.

Task 2

1. Data Preprocessing

I choose to use standardization rather than normalization to preprocess the data set (see mfile `preprocess.m`), there are some reasons:

(a) Sensitivity of SVM to Feature Scales

Support Vector Machines (SVMs), particularly those employing nonlinear kernels such as Radial Basis Function (RBF) or polynomial kernels, are inherently sensitive to feature scaling. This sensitivity arises because the distance metrics used in kernel computations are directly influenced by the magnitude of features. Features with larger scales dominate the kernel calculations, leading to biased model outputs. Standardization mitigates this issue by ensuring all features contribute equally to the distance metric.

(b) Preservation of Relative Variation Trends

Unlike min-max normalization (rescaling to [0, 1]), standardization preserves the relative variation trends between features. This is critical for maintaining the interpretability of feature importance and avoiding distortion of the original data distribution. Min-max normalization, in contrast, compresses all features into a fixed range, which may amplify the impact of outliers and obscure natural variations.

(c) Enhanced Training Convergence and Generalization

By transforming features to zero mean and unit variance, standardization stabilizes gradient-based optimization processes. This accelerates convergence and reduces the risk of numerical instability. Furthermore, standardized features improve model generalization by aligning the training and test data distributions, which is essential for SVMs to achieve robust performance on unseen data.

2. Admissibility of Kernels and Existence of Optimal Hyperplanes

(a) Admissibility of Kernels

Theoretical guarantee: Both the linear kernel and the polynomial kernels $(x^T y + c)^p$ are proven to be valid Mercer kernel for all $p \geq 1, c \geq 0$, as established in literature[1][2][3].

Experimental observation: Numerical instability in high-degree ($p \geq 3$) polynomial kernels leads to ill-conditioned Gram matrices. The observed small negative eigenvalues arise from floating-point errors, not theoretical invalidity.

Solutions: I use relative tolerance (tol=(1e-12)*max_eigenvalue) to solve the numerical error problem (see mfile `validate_mercer_condition.m`).

(b) Existence of Optimal Hyperplanes

For **hard-margin** SVMs, a hyperplane exists if and only if the data is linearly separable in the feature space induced by the kernel. For $p = 5$, the hard-margin SVM failed to converge ('`quadprog`' function return `exitflag=0`, which means number of iterations exceeded `options.MaxIterations`), indicating that the data is inseparable in the high-dimensional space. This implies that high-dimensional mappings (e.g., p=5) do not guarantee separability. Overly complex kernels risk overfitting and algorithmic instability.

By introducing slack variables ξ_i, **soft-margin** SVMs tolerate misclassifications, ensuring a feasible solution even for inseparable data.

(c) Conclusion and Key Insight

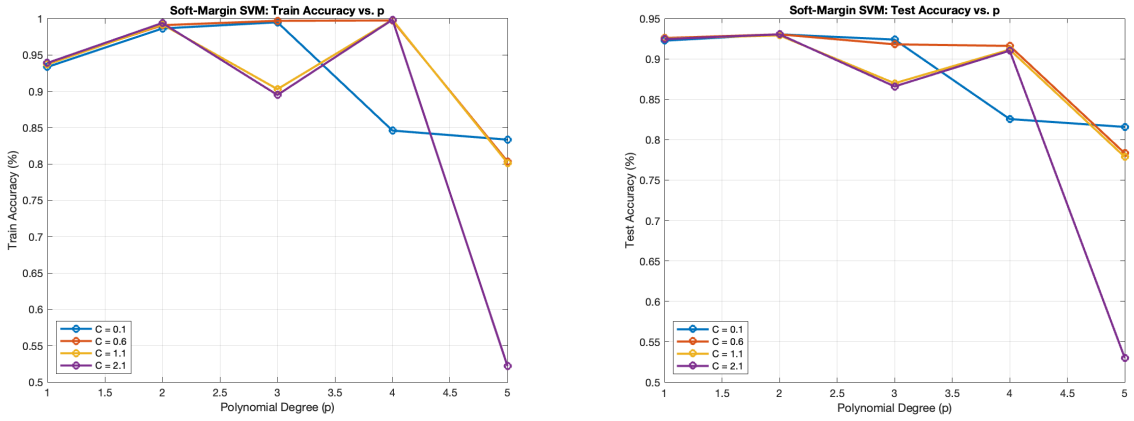
While both hard- and soft-margin SVMs (with supporting argument) rely on admissible kernels, their practical utility diverges sharply due to the data separability. The hard-margin formulation's rigidity renders it unsuitable for most real-world scenarios, whereas soft-margin SVMs, with judicious C-tuning, offer robust performance even with complex kernels. This project underscores the necessity of balancing theoretical kernel validity with empirical stability and generalization.

3. Experiment Results

I implement training a SVM in a mfile `train_svm.m`, using "quadprog" to solve the constraint optimization problem. The results are shown in Table 1. The complete procedure of training SVMs and prediction (Task 1 and Task 2) is in the mfile `main_workflow.m`.

Type of SVM	**Training accuracy**				**Test accuracy**			
Hard margin with Linear kernel	93.95%				92.77%			
Hard margin with polynomial kernel	p=2	p=3	p=4	p=5	p=2	p=3	p=4	p=5
	99.85%	81.95%	74.50%	54.05%	**91.02**%	78.19%	72.46%	53.06%
Soft margin with polynomial kernel	C=0.1	C=0.6	C=1.1	C=2.1	C=0.1	C=0.6	C=1.1	C=2.1
p=1	93.35%	93.85%	93.70%	93.90%	92.25%	92.58%	92.51%	92.45%
p=2	**98.65**%	99.10%	99.20%	99.40%	**93.03**%	93.03%	92.90%	93.03%
p=3	99.50%	99.70%	90.30%	89.50%	92.38%	91.80%	86.98%	86.59%
p=4	84.60%	99.75%	99.85%	**99.85**%	82.55%	91.60%	91.15%	91.02%
p=5	83.35%	80.35%	80.15%	52.20%	81.58%	78.32%	77.86%	52.99%

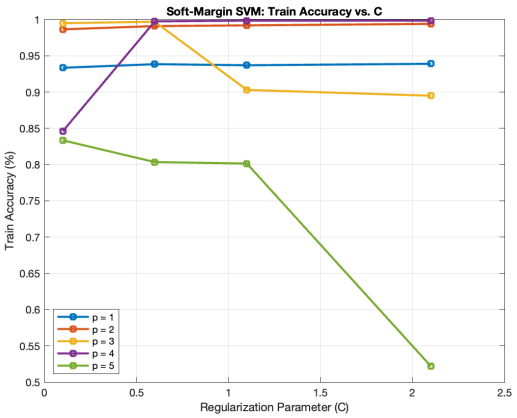
Table 1: Results of SVM classification



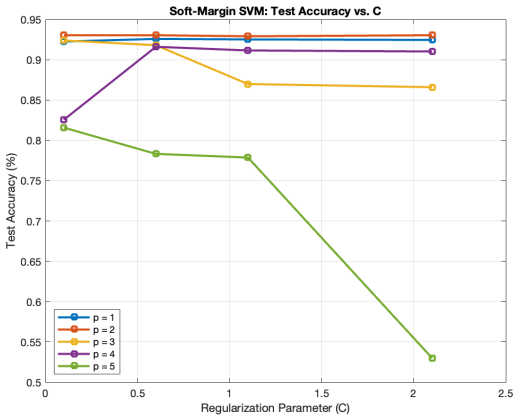
(a) Train Accuracy vs. p

(b) Test Accuracy vs. p

Figure 1: Performance of Soft-Margin SVMs vs. p

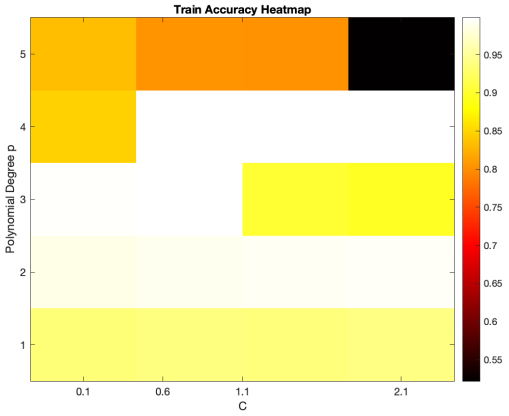


(a) Train Accuracy vs. C

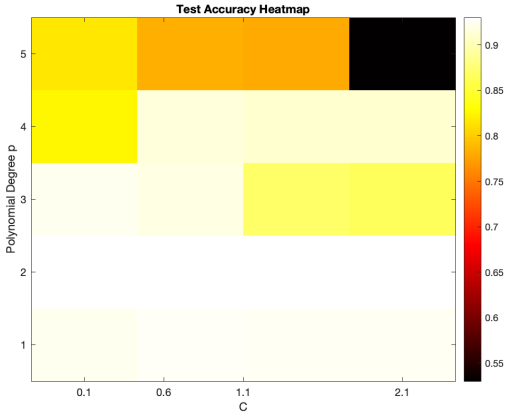


(b) Test Accuracy vs. C

Figure 2: Performance of Soft-Margin SVMs vs. C



(a) Train Accuracy Heatmap



(b) Test Accuracy Heatmap

Figure 3: Accuracy Heatmap of Soft-Margin SVMs

3. Comments on Results

(a) Comparison of Hard-Margin and Soft-Margin SVMs

The **linear** kernel, the simplest kernel function, is suitable for linearly separable data. The **polynomial** kernel allows for more complex decision boundaries by adding polynomial features to the data.

(i) Hard-Margin SVMs rely on strict data separability and are highly sensitive to outliers and noise. The linear kernel achieves a training accuracy of **93.95**% and a test accuracy of **92.77**%, indicating that the dataset is nearly linearly separable. However, the linear kernel fails to capture more complex nonlinear patterns.

When using a polynomial kernel, moderate degrees (e.g., $p = 2$) significantly improve training accuracy to **99.85**%, demonstrating the kernel's ability to model nonlinear relationships via polynomial expansion. Nonetheless, higher degrees (e.g., $p = 5$) result in severe overfitting, where both training and test accuracy deteriorate sharply (e.g., test accuracy drops to 53.06%). This highlights the trade-off between model flexibility and generalization.

(ii) Soft-Margin SVMs introduce **regularization** to balance margin width and classification errors, which enhances robustness to noise and improves generalization. For instance, using a polynomial kernel with $p = 2$ and moderate regularization ($C = 0.6$) yields a test accuracy of **93.03**, outperforming the corresponding hard-margin counterpart (91.02%). The ability to adjust regularization via the parameter C allows soft-margin SVMs to maintain good performance even when the data is not perfectly separable. The tolerance for misclassifications reduces sensitivity to noise and outliers, enabling better generalization on unseen data.

Aspect	**Hard-Margin SVM**	**Soft-Margin SVM**
Feasibility	Requires perfect separability	Always feasible due to slack variables
Sensitivity to Noise	Highly sensitive to noise	Robust to noise via C-controlled misclassification
Generalization	Prone to overfitting	Better generalization with proper C tuning
Numerical Stability	Vulnerable to ill-conditioned kernels	Regularization (via C) improved stability
Use Case	Theoretical or ideal datasets with guaranteed separability	Practical datasets with noise or complexity

Table 2: Comparative Summary: Hard-Margin vs. Soft-Margin

(b) Synergy Between Kernel Complexity and Regularization

The regularization parameter C controls the trade-off between minimizing training errors and maximizing the margin. Its interaction with kernel complexity p significantly influences SVM performance.

(i) Small C Value and Large Margin: A small C encourages a wider margin by allowing more training errors. This can be useful when the data points are well-separated, and there is a low presence of noise or outliers.

(ii) Large C Value and Narrow Margin: A large C imposes heavy penalties on misclassifications, favoring narrow margin and perfect training performance, but increasing the risk of overfitting.

(iii) Performance Trends and Optimization Strategies

For low-complexity kernels ($p \leq 2$), increasing C slightly improves training accuracy, while test accuracy remains stable ($p = 2$, 93.03% across $C = 0.1 to 2.1$), approximating to behavior of a hard-margin SVM as $C \to \infty$.

For high-complexity kernels ($p \geq 3$), small C values (≤ 1) help mitigate overfitting by tolerating misclassification of noisy samples. For example, with $p = 3$ and $C = 0.1$, the test accuracy is 92.38%, compared to 86.59% at $C = 2.1$. However, excessively small C ($\ll 1$) may lead to underfitting, where the model fails to capture the important patterns (e.g., $p = 4, C = 0.1$: training=84.60%, test=82.55%; $C = 2.1$: training=99.85%, test=91.02%), degrading both training and test accuracy. Increasing C forces the excessively complex model ($p = 5$) to fit the outliers thus degrades the performance significantly (also see Figure 2).

(iv) Summary

For moderate kernel complexity ($p = 2$), using an intermediate regularization value (e.g., $C = 0.6$) achieves the highest test accuracy (as shown in Figure 3). For higher-degree kernels ($p \geq 3$), the model becomes increasingly prone to overfitting, and stronger regularization (smaller $C \leq 1$) is essential to preserve generalization. Although increasing p improves training accuracy, it does not guarantee better performance on unseen data, which indicates the model is fitting noise rather than useful patterns. Therefore, extreme kernel degrees ($p > 4$) should be avoided unless justified by the inherent nonlinearity of the data.

Task 3 Design a SVM

1. Design Objective

The objective of this task is to design an optimized Support Vector Machine (SVM) model for classifying spam emails, based on insights from previous experiments with various kernels and regularization settings. The final design aims to maximize generalization performance on unseen data while maintaining high training accuracy and numerical stability.

2. Design Choice

Given the previous evaluation of kernel functions, the Radial Basis Function (RBF) kernel was selected for the final design. RBF kernels are known for their ability to capture complex, non-linear patterns through localized decision boundaries.

The feature vectors are standardized to zero mean and unit variance. Dimensionality reduction techniques such as Principal Component Analysis (PCA) were explored but ultimately excluded from the final design due to decreased test performance (see Section 5).

3. Hyperparameter Optimization

To tune the RBF kernel, three grid searches were performed over the following ranges:

(a) Round 1:

- Kernel scale parameter $\gamma \in 0.001, 0.01, 0.1, 1$
- Regularization parameter $C \in 0.1, 1, 10, 100$

(b) Round 2:

- $\gamma \in 0.01, 0.02, 0.04, 0.06, 0.08$
- $C \in 10, 20, 30, 40, 50, 60, 70, 80, 90$

(c) Round 3:

- $\gamma \in 0.03, 0.035, 0.04, 0.045, 0.05$
- $C \in 35, 40, 45, 50, 55$

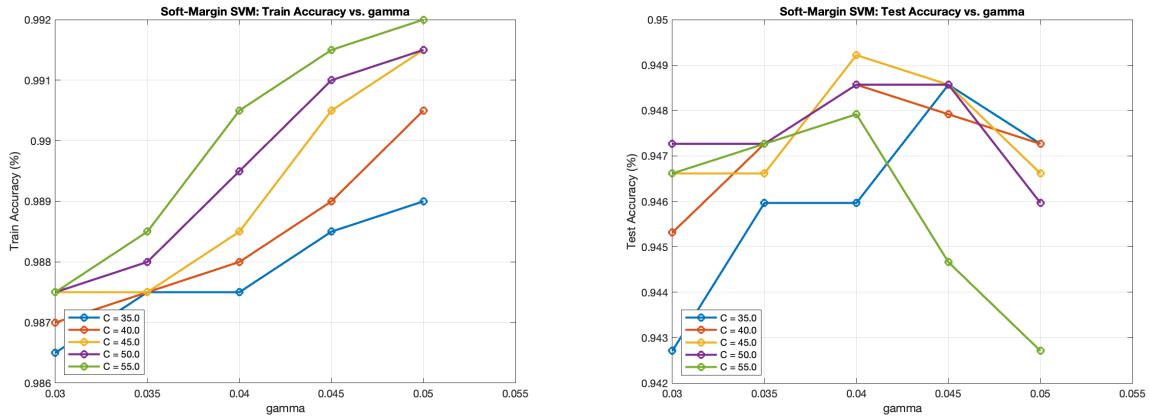
After extensive evaluation, the optimal configuration was found at: $\gamma = 0.04$, $C = 45$.

This combination yielded the best overall test performance: Training accuracy: 98.85%, Test accuracy: 94.92%. This outperformed the previously best-performing polynomial kernel

configuration ($p = 2, C = 0.6$), which achieved 99.10% training accuracy but only 93.03% test accuracy.

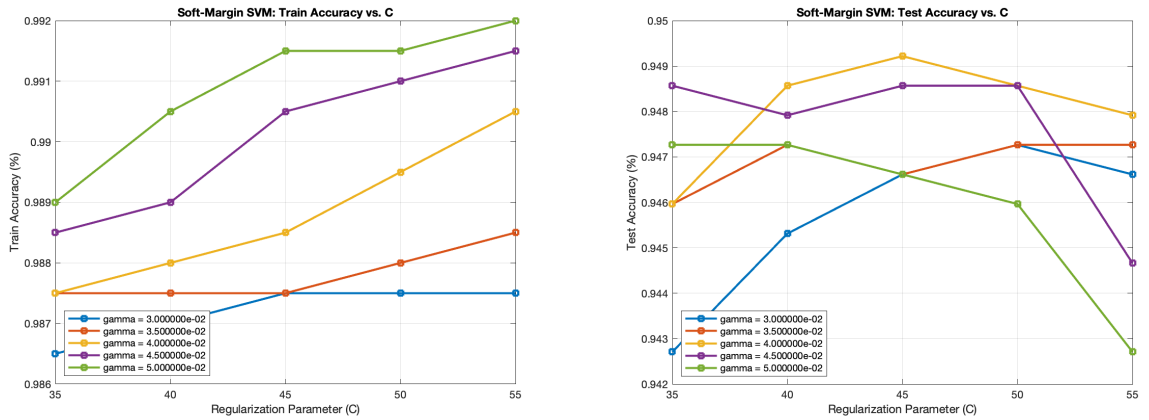
	Training accuracy					**Test accuracy**				
	C=35	C=40	**C=45**	C=50	C=55	C=35	C=40	**C=45**	C=50	C=55
γ=0.03	98.65%	98.70%	98.75%	98.75%	98.75%	94.27%	94.53%	94.66%	94.73%	94.66%
γ=0.035	98.75%	98.75%	98.75%	98.80%	98.85%	94.60%	94.73%	94.66%	94.73%	94.73%
γ=**0.04**	98.75%	98.80%	**98.85**%	98.95%	99.05%	94.60%	94.86%	**94.92**%	94.86%	94.79%
γ=0.045	98.85%	98.90%	99.05%	99.10%	99.15%	94.86%	94.79%	94.86%	94.86%	94.47%
γ=0.05	98.90%	99.05%	99.15%	99.15%	99.20%	94.73%	94.73%	94.66%	94.60%	94.27%

Table 3: Classification Performance of the Designed SVM



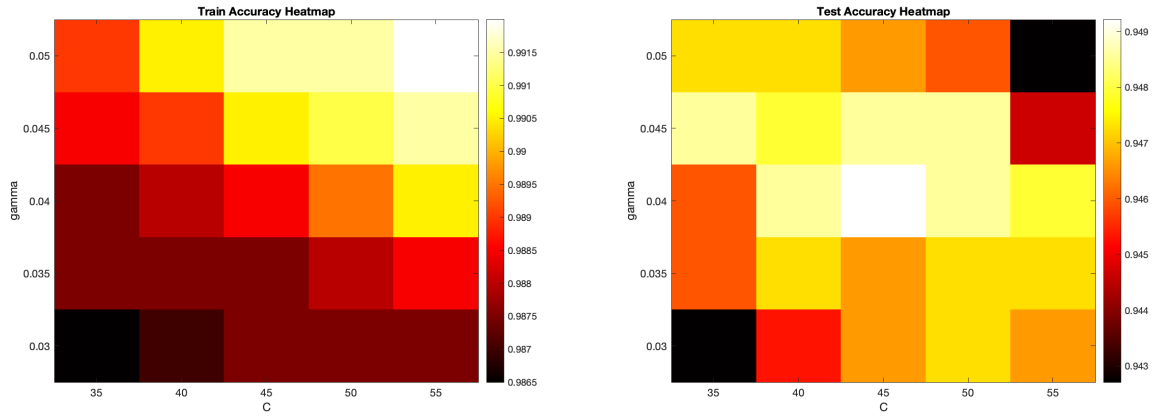
(a) Train Accuracy vs. γ (b) Test Accuracy vs. γ

Figure 4: Performance of Soft-Margin SVMs vs. γ



(a) Train Accuracy vs. C (b) Test Accuracy vs. C

Figure 5: Performance of Soft-Margin SVMs vs. C



(a) Train Accuracy Heatmap

(b) Test Accuracy Heatmap

Figure 6: Accuracy Heatmap of Soft-Margin SVMs

4. Performance Comparison and Discussion

The RBF kernel offers improved generalization by enforcing localized similarity measures, which better handle complex, non-linear relationships in the dataset. While the polynomial kernel achieved slightly higher training accuracy, it exhibited mild overfitting, as evidenced by the test accuracy gap.

In contrast, the RBF kernel SVM maintained a strong balance between bias and variance, leading to improved robustness. The results confirm that optimizing both kernel shape and regularization strength is critical for model generalization.

5. PCA Exploration and Its Limitation

As part of the investigation, PCA was applied to reduce the input feature space from 57 dimensions to 47 dimensions, preserving approximately 95% of the total variance. However, when training the SVM on the PCA-transformed data, no configuration achieved higher test accuracy than the original 57-dimensional input.

This may be attributed to the unsupervised nature of PCA, which selects components based on variance rather than discriminative power. In this case, relevant classification features may lie in low-variance directions, and dimensionality reduction could inadvertently remove them. As a result, PCA was excluded from the final pipeline.

6. Final Conclusion

The final model is an SVM with an RBF kernel, trained on the original 57-dimensional standardized features, using hyperparameters $\gamma = 0.04$ and $C = 45$. This configuration achieved the highest generalization performance in all experiments, with a test accuracy of 94.92%. The model demonstrates a strong ability to capture non-linear patterns while maintaining robustness and numerical stability, and is therefore selected as the final design.

Appendix

1. SVM Optimization Problem

(a) Hard-margin SVM

$$\text{Minimize: } \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{Subject to: } d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \forall i$$

In the hard margin SVM setting, the goal is to find the hyperplane that separates the data classes, assuming they are linearly separable.

(b) Soft-margin SVM

$$\text{Minimize: } \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

$$\text{Subject to: } d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i \geq 0, \xi_i \geq 0$$

In soft margin SVM, slack variables ξ_i, are introduces to allow some margin violations. Finite C allows misclassification.

2. Dual Problem

(a) Hard-margin SVM
Define Lagrangian function:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1]$$

Set gradient to zero:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N}\alpha_i d_i \mathbf{x}_i = 0, \quad \frac{\partial L}{\partial b} = -\sum_{i=1}^{N}\alpha_i d_i = 0$$

Substitute back to get the dual problem: Find: α_i,

$$\text{Maximize: } Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

$$\text{Subject to: } \sum_{i=1}^{N}\alpha_i d_i = 0, \quad \alpha_i \geq 0$$

$\mathbf{x}_i^T\mathbf{x}_j$ is called a linear kernel.

(b) Soft-margin SVM (with transformation)
Similarly, using Lagrangian function and the KKT conditions, we can get the dual problem for soft-margin SVM: Find: α_i,

$$\text{Maximize: } Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j \varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_j)$$

$$\text{Subject to: } \sum_{i=1}^{N} \alpha_i d_i = 0, \quad 0 \leq \alpha_i \leq C$$

$\varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_j)$ is kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, when using polynomial kernel, it becomes $(\mathbf{x}_i^T \mathbf{x}_j + 1)^p$.

This is a convex quadratic programming (QP) problem, once $\alpha_{\circ,i}$ is solved, we can calculate $\mathbf{w}_\circ$ as follow:

$$\mathbf{w}_\circ = \sum_{i=1}^{N} \alpha_{\circ,i} d_i \mathbf{x}_i$$

For support vector (SV) $\mathbf{x}_i$ with $\alpha_i > 0$ (hard margin) or with $0 < \alpha_i \leq C$ (soft margin), the bias $b_\circ$ can be computed from the support vectors $\mathbf{x}_i$ using:

$$b_{\circ,i} = \frac{1}{d_i} - \mathbf{w}_\circ^T \mathbf{x}_i$$

Take $b_\circ$ as the average of all such $b_{\circ,i}$,

$$b_\circ = \frac{\sum_{i=1}^{m} b_{\circ,i}}{m}$$

where m is the total number of $\mathbf{x}_i$ with $0 < \alpha_i \leq C$.

References

References

[1] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006.

[2] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[3] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.