EE5904/ME5404 Neural Networks AY2024/2025 Semester 2

SVM
for Classification of Spam Email Messages

XU NUO

A0313771H

e1499166@u.nus.edu

April 7, 2025

1. Data Pre-processing

Before constructing the support vector machine (SVM), it is necessary to preprocess both the training and test datasets. The demo provides two preprocessing methods: normalization and standardization.

Here, I adopted the standardization method, which transforms each feature dimension to have a mean of 0 and standard deviation of 1. The expressions related to standardization are as follows:

$$Mean:\ \ \mu_i = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$Standard\ Deviation:\ \ \sigma_i = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_i)^2}$$

$$Standardization:\ \ x_i' = \frac{x_i - \mu_i}{\sigma_i}$$

where μ_i and σ_i are computed from the training set only.

To avoid division by zero for features with no variance, I replaced near-zero standard deviations with 1:

$$\sigma_i = 1\ for\ \sigma_i < 1e-10$$

This ensures numerical stability and consistent behavior during optimization.

Standardizing the data helps eliminate differences in feature scales. Different features may have different units or magnitudes, and such discrepancies can affect the computation of inner products. Standardization also facilitates faster convergence of the weight parameters during optimization and improves the generalization ability of the model.

2. SVM Implementation

2.1 Explanation of Principle

Consider the following training dataset:

$$\{(x_1, d_1), (x_2, d_2), \dots, (x_n, d_n)\},\ \ x_i \in \mathbb{R}^d,\ \ d_i \in \{-1, +1\}$$

Our goal is to find a hyperplane to separate the positive and negative classes, in the form:

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0$$

The distance from a point $\boldsymbol{x}$ to the hyperplane is:

$$Distance = \frac{|\boldsymbol{w}^T \boldsymbol{x} + b|}{||\boldsymbol{w}||}$$

In the context of SVM, we require that all training samples are correctly classified and lie at least a distance of 1 from the hyperplane, which gives the constraint:

$$d_i(\boldsymbol{w}^T x_i + b) \geq 1 \quad for\ all\ i$$

Therefore, the distance between the closest points from both classes is:

$$Margin = \frac{2}{||\boldsymbol{w}||}$$

The objective is to maximize the margin under the condition that all samples are correctly classified. For ease of computation, this is equivalent to minimizing one-half of the squared norm of $\boldsymbol{w}$. This leads to the Primal Form of the SVM, which is expressed as follows:

$$\min_{w,b} \frac{1}{2}||w||^2, \ \ subject\ to: \ \ d_i(\boldsymbol{w}^T x_i + b) \geq 1$$

We construct the Lagrangian function as follows:

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i [d_i(\boldsymbol{w}^T x_i + b) - 1]$$

By taking partial derivatives of L with respect to $\boldsymbol{w}$ and b, and setting them to zero, we eliminate $\boldsymbol{w}$ and b from the expression and obtain the Dual Problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j, \ \ subject\ to: \ \ \sum_{i=1}^{n} \alpha_i d_i = 0, \ \ \alpha_i \geq 0$$

By replacing the inner product $x_i^T x_j$ with a kernel function $K(x_i, x_j)$, we can achieve nonlinear classification in a high-dimensional feature space and avoid explicitly computing the high-dimensional mappings. Finally, we turn this into solving following QP problem:

$$\min_{\alpha} \frac{1}{2} \alpha^T H \alpha - \mathbf{1}^T \alpha, \ \ subject\ to: \ \ \alpha_i \geq 0, \sum \alpha_i d_i = 0, \ \ where\ H_{ij} = d_i d_j K(x_i, x_j)$$

The formulations listed above are merely the basic expressions of the dual problem. Depending on the specific kernel function and misclassification tolerance, the parameters and implementation will vary. The specific optimization constraints are described in the following sections.

Once we solve the dual problem and obtain the optimal Lagrange multipliers α_i, we

can construct the optimal hyperplane. The discriminant function is used to classify a new input $\boldsymbol{x}$ based on the following expression:

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i d_i K(x_i, \boldsymbol{x}) + b$$

If $g(\boldsymbol{x}) > 0$, we predict its label as +1, -1 if not.

To compute the bias b, we select any support vector x_k and use KKT condition:

$$d_k \left(\sum_{i=1}^{n} \alpha_i d_i K(x_i, x_k) + b \right) = 1$$

$$b = d_k - \sum_{i=1}^{n} \alpha_i d_i K(x_i, x_k)$$

2.2 Quadprog Function and General Parameter settings

The key function to solve this problem is the *quadprog* function in the Optimization Toolbox in MATLAB:

$$[alpha, fval, exitflag] = quadprog(H, f, A, b, Aeq, beq, lb, ub, x0, options)$$

which is a minimization problem specified by

$$\min_x \frac{1}{2} x^T H x + f^T x \ \ such\ that \begin{cases} A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases}$$

The general parameter setting involved in *quadprog* in the code is as follows:

```
N = size(data_train, 2); % Get the number of samples of train_data
f = -ones(N, 1); % Predefine for later use of the linear term "-sum(ai)" (function to
minimize "0.5a' * H * a -sum(a_i)")
A = []; b = []; % No inequality constraints for SVM (meaning no Ax <= b constraint)
Aeq = label'; % Equality constraint Aeq * a = beq
beq = 0; % sum(a_i * y_i) = 0 ensures "the force of the two types of the samlpes on the
hyperplane is the same"
lb = zeros(N, 1); % Lower bound: a_i >= 0 (Lagrange multiplier)
ub = C * ones(N, 1); % Upper bound
x0 = []; % No initial point given
H = label * label' .* K_train; % Define an N*N symmetric matrix H_ij = y_i * y_j' .*
K(x_i, x_j) for the dual form of SVM
H = (H + H') / 2; % To ensure symmetry and positive definiteness (to avoid numerical
errors), force H to be symmetric
options = optimoptions('quadprog', 'Display', 'Iter', 'MaxIterations', 1000);
```

```
[alpha, ~, exitflag] = quadprog(H, f, A, b, Aeq, beq, lb, ub, x0, options); % Use quadprog and display iteration information
```

2.3 Hard Margin with Linear Kernel

Hard margin SVM is the most fundamental and original form of SVM. It assumes that the training data is linearly separable and does not tolerate any classification errors.

Using a linear kernel means that the data is not mapped into a higher dimensional space; instead, a linear hyperplane is directly found in the original feature space for classification. The corresponding kernel function is:

$$K(x_i, x_j) = x_i^T x_j$$

The optimization objective function and constraints in this SVM dual form are as follows:

$$Maximize:\ \ Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j$$

$$Subject\ to:\ \sum_{i=1}^{n} \alpha_i d_i = 0, \qquad \alpha_i \geq 0$$

2.4 Hard Margin with Polynomial Kernel

Using a polynomial kernel means that the data is mapped into a higher dimensional space. It introduces high-order feature interaction terms to make non-linearly separable data become linearly separable in high-dimensional space. The corresponding kernel function is:

$$K(x_i, x_j) = (x_i^T x_j + r)^p$$

where r is the bias term, typically set to 1, p is the degree of the polynomial.

The optimization objective function and constraints in this SVM dual form are as follows:

$$Maximize:\ \ Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j (x_i^T x_j + 1)^p$$

$$Subject\ to:\ \sum_{i=1}^{n} \alpha_i d_i = 0, \qquad \alpha_i \geq 0$$

2.5 Soft Margin with Polynomial Kernel

By introducing slack variables $\xi_i \geq 0$ and adding a regularization parameter C, soft margin SVM allows some samples to be misclassified to improve the model's tolerance and generalization ability to noise and nonlinearly separable data. Large C heavily penalizes misclassification (fewer classification errors but risk of overfitting) and vice versa. The objective function becomes:

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

Subject to the constraints:

$$d_i(\boldsymbol{w}^T x_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

The optimization objective function and constraints in this SVM dual form are as follows:

$$Maximize: \ Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j \left(x_i^T x_j + 1\right)^p$$

$$Subject\ to: \ \sum_{i=1}^{n} \alpha_i d_i = 0, \qquad 0 \leq \alpha_i \leq C$$

3. Admissibility of the Kernels

To determine whether a kernel function is admissible, it is necessary to verify whether it satisfies Mercer's condition. For a symmetric function $K(x, y)$, it satisfies Mercer's condition if and only if the following holds for any function $f(x)$:

$$\iint f(x)K(x,y)f(y)dxdy \geq 0$$

This means that the kernel matrix defined by the kernel function, $K_{ij} = K(x_i, x_j)$ mut be positive semi-definite. The Gram matrix must satisfy $\alpha^T K\alpha \geq 0$ for any vector α. The positive semi-definiteness of a matrix can be verified by checking whether all its eigenvalues are greater than or equal to zero.

$$Eigenvalues\ of\ K(x_i, x_j) > -1e-4$$

Due to floating-point precision limitations in computer calculations, we adopt a numerical tolerance of -1×10^{-4}. As long as all eigenvalues are greater than -1×10^{-4}, the kernel function is considered admissible.

After computation and evaluation in MATLAB, we can conclude that when using the polynomial kernel, the kernel is inadmissible for both hard margin and soft margin

cases when $p = 4$ and $p = 5$. In all other cases required by the task, the kernel functions used are admissible.

4. Existence of Optimal Hyperplanes

I determine the existence of an optimal hyperplane based on two conditions. The first is to check whether the Lagrange multipliers α_i corresponding to the support vectors obtained during training are non-empty. The second is to examine whether the solver *quadprog* returns a positive exit flag. If both conditions are satisfied, the model is considered to have an optimal separating hyperplane.

According to the program output, the only case where no optimal hyperplane exists is when using a hard-margin SVM with a polynomial kernel of degree $p = 5$. The Command Window displays that *solver stopped prematurely, quadprog stopped because it exceeded the iteration limit, options.MaxIterations = 1.000000e+03*. In all other cases, it displays that *minimum found that satisfies the constraints. Optimization completed because the objective function is non-decreasing in feasible directions, to within the value of the optimality tolerance, and constraints are satisfied to within the value of the constraint tolerance.*

5. Comments on Results

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%	99.9%	95.1%	91.55%	91.02%	90.43%	86.46%	85.29%
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%	93.7%	93.9%	92.25%	92.58%	92.51%	92.44%
P=2	98.65%	99.1%	99.15%	99.3%	92.97%	93.03%	92.90%	93.10%
P=3	98.55%	98.3%	97.6%	97.7%	91.80%	91.28%	90.23%	90.36%
P=4	91.15%	92.25%	91.4%	91.6%	84.44%	84.64%	83.40%	83.33%
P=5	87.45%	85.95%	86.85%	86.95%	81.97%	81.45%	81.38%	80.86%

TABLE 1: Results of SVM Classification

For reference, the prediction results obtained with inadmissible kernels ($p = 4/5$) are also shown here. Based on the results shown in Table 1, we can make the following comments:

Performance of Linear vs. Polynomial Kernels

The hard margin SVM with a linear kernel achieves 93.95% training accuracy. While relatively good, it is outperformed by polynomial kernels with suitable degrees (e.g., $p = 2/3$), which capture more complex decision boundaries. This highlights the benefit of non-linear kernels in handling non-linearly separable data. However, the corresponding test accuracies are only 91.02% and 90.43%, which are lower than the 92.77% achieved by the linear kernel. This suggests that the hard margin's intolerance to classification errors leads to overfitting.

Effect of Polynomial Degree p:

The train and test accuracy initially improve with increasing polynomial degree. However, at higher degrees ($p = 4/5$), performance drops significantly. This decrease is largely due to the violation of the Mercer condition, making the kernel inadmissible and compromising the models' capacity to learn valid hyperplanes.

Soft Margin vs. Hard Margin SVM:

Soft margin SVM generally outperforms hard margin SVM in test accuracy under the same kernel. For example, with $p = 2$, the test accuracy is 91.02% using hard margin compared to 93.10% using soft margin ($C = 2.1$). This confirms the advantage of allowing small classification errors, especially when data are not perfectly linearly separable.

Effect of Soft Margin Parameter C:

A smaller C value (e.g., 0.1) provides stronger regularization and typically leads to lower training accuracy but better generalization. As C increases (e.g., 2.1), training accuracy improves, but the risk of overfitting also rises.

The results suggest that $C = 2.1$ with $p = 2$ provides the best balance between fitting and generalization.

6. Discussion on Design Decisions

In this section, I designed an SVM using a soft margin with an RBF kernel. The gaussian kernel takes the following form:

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

where $\gamma = \frac{1}{2\sigma^2}$ controls the width of the kernel.

The corresponding dual optimization problem is formulated as follows:

$$Maximize:\ Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j e^{-\gamma\left||x_i - x_j|\right|^2}$$

subject to:

$$\sum_{i=1}^{n} \alpha_i d_i = 0, \qquad 0 \leq \alpha_i \leq C$$

γ determines the influence range of each training sample in the feature space. A larger γ results in a more localized influence, causing the model to vary sharply around each sample, which can easily lead to overfitting. Conversely, a smaller γ leads to a broader influence range, making the model smoother and more generalized, but also more prone to underfitting.

According to the above principles, I ultimately chose $\sigma = 10$, $\gamma = 0.005$ and $C = 1000$. The accuracy of this SVM on the training set, test set, and dummy evaluation set is shown in Table 2. The dummy evaluation set consists of 600 samples randomly selected from the test set and is used to verify the reliability of the code implementation.

Train Accuracy	Test Accuracy	Dummy Eval Accuracy
98.60%	94.27%	94.00%

TABLE 2: Results of SVM Classification Using Soft Margin with an RBF Kernel