Faculty of Engineering

Name	Zavier Ong Jin Jie
Matriculation No.	A0138993L
Email	E0002878@u.nus.edu
Module	EE5904
Project Title	SVM for classification of Spam Email Messages

Data Pre-processing

Based on the SVM_Project_demo, we would need to pre-process the data by with our own method of choice. In this project, the pre-processing method chosen would be the Standardization method.

To standardize the data, each feature would be transformed by removing the mean value of each feature followed by dividing by each feature's standard deviation as visualized below in equation 1:

$$z = \frac{(x - \mu)}{\sigma}$$

Equation 1. Standardization Method

Kernel Suitability

For all SVMs in this report, we will use the Mercer's condition to check for the kernel suitability. The Mercer's condition is defined as: For a training set $S = \{(x_i, d_i)\}, i = 1,2, \dots, N$, the Gram matrix (K) is positive semi-definite, meaning that the eigenvalues for the gram matrix are nonnegative.

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_N, \mathbf{x}_1) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \in R^{N \times N}$$

Figure 1. Gram matrix

Task 1: Compute discriminant function

For this task, we are required to compute the discriminant function $g(\cdot)$, if it exists. We can make use of the *quadprog* toolbox in MATLAB to calculate α_i. The *quadprog* function is a solver for quadratic objective functions with linear constraints to find a minimum for a problem specified in Figure 2:

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

Figure 2. quadprog function

Where *H, A,* and *Aeq* are matrices while *f, b, beq, lb, ub* and *x* are vectors.

Hard margin Linear SVM

To know the discriminant function $g(\cdot)$ exists, we would first need to generate the gram matrix. The gram matrix for a hard margin SVM with a linear kernel is defined as:

$$K(x_1, x_2) = {x_1}^T x_2$$

Equation 2. Gram matrix for linear kernel

while C is set to 10^6 to simulate $+\infty$ in theory. α_i can be calculated in the Figure 3 below following the Lagrange multiplier constraint of $0 \leq \alpha_i$:

```
H = K.*(train_label*train_label');
f = -ones(2000, 1);
A = [];
b= [];
Aeq = train_label';
beq = 0;
lb = zeros(2000, 1);
ub = ones(2000, 1)*C;
x0 = [];
options = optimset('LargeScale','off','MaxIter', 1000);
%calc alpha - quadprog
alpha = quadprog(H, f, A, b, Aeq, beq, lb, ub, x0, options);
```

Figure 3. Calculating α_i

Th discriminant function and its discriminant parameters for a linear kernel is defined as follows in equation 3 and Figure 4 below:

$$g(x) = {w_o}^T x + b_0$$

Equation 3. Discriminant function for linear kernel

$$\mathbf{w}_\circ = \sum_{i=1}^{N} \alpha_{\circ,i}\, d_i\, \mathbf{x}_i\,, \quad b_\circ = \frac{1}{d^{(s)}} - \mathbf{w}_\circ^T \mathbf{x}^{(s)}$$

$x^{(s)} \rightarrow$ support vector
$d^{(s)} \rightarrow label$

Figure 4 Discriminant parameters for linear kernel

To select support vectors, we choose a threshold of 1e-4.

Polynomial SVM

Like hard-margin linear SVM, the steps employed are similar except the gram matrix (K) and discriminant terms. The gram matrix of a polynomial kernel is defined below:

$$K(x_1, x_2) = {(x_1}^T x_2 + 1)^p$$

Equation 4. Gram matrix for polynomial kernel

Like Figure 3, α_i is calculated in the same way but with varying C. The discriminant function and parameters for a nonlinear (polynomial) kernel is defined in equation 5 and figure 5 below:

$$g(x) = \sum_{i=1}^{N} a_{o,i} d_i K(x_1, x_2) + b_0$$

Equation 5. Discriminant function for nonlinear kernel

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

using the fact that for a support vector $\mathbf{x}^{(s)}$

$$g(\mathbf{x}^{(s)}) = \pm 1 = d^{(s)}$$

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 $0 < \alpha_i \leq C$.

Figure 5. Discriminant parameters for nonlinear kernel

Hard margin

In the case of hard margin nonlinear kernel, we would be selecting support vectors that fulfil the Lagrange multiplier constraint of $0 \leq \alpha_i \leq 10^6$.

Soft margin

In the case of soft margin nonlinear kernel, we would select support vectors that fulfil the constraint on the Lagrange multiplier where $0 \leq \alpha_i \leq C$ where C = (0.1, 0.6, 1.1 2.1).

Task 2: Implementation of SVMs

Classification Results

Type of SVM	Training Accuracy (%)				Test Accuracy (%)			
Hard margin with linear kernel	92.95				91.54			
Hard margin with polynomial kernel	P=2	P=3	P=4	P=5	P=2	P=3	P=4	P=5
	100	100	42.45	41.25	85.87	85.94	40.43	39.32
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	92.65	92.50	92.65	92.90	92.97	92.71	92.97	93.23
P = 2	98.90	99.50	99.50	99.55	89.39	89.45	88.48	88.09
P = 3	99.65	99.80	99.85	99.90	89.13	88.48	88.35	88.35
P = 4	99.90	99.95	100	100	87.89	86.78	86	86
P = 5	98.90	98.90	98.95	99.15	87.30	86.39	86	86.13

Table 1. Results of SVM classification

Discuss results and implications.

In table 1, not admissible kernels results are coloured red. Even though they are still capable of yielding results, its test accuracies are generally worse compared to admissible kernels. Therefore, we should not be considering parameters when the kernels are not admissible.

In addition to that, when making use of the *quadprog* toolbox in MATLAB, there are cases when the *quadprog* function would return “The problem is non-convex”, which would mean that it is unable to maximize the width of the margin. This suggests that an optimal hyperplane does not exist in this configuration of SVM. On the other hand, if the *quadprog* function is able to find a minimum that fits within the constraints of the upper and lower bound, it would suggest that there exists a optimal hyperplane to linearly separate the data.

By comparing both hard margin kernels, the linear kernel clearly outperforms the polynomial kernel. This could suggest that a hard margin polynomial kernel is overfitting to the training dataset.

With a soft margin polynomial kernel, we can see that test accuracies are all better as compared to a hard margin polynomial kernel. This is because a soft margin would allow some data to enter the margin intentionally to prevent overfitting. However, we would still penalize that points. The parameter C represents how much we would penalize the SVM for a misclassification for a data point.

In soft margin with polynomial SVM, we can observe that at high values of C, the SVM tend to overfit more towards the training data, especially at higher values of P. Therefore, small values of P and C seems to be able to produce better test accuracies.

Task 3: Design own SVM

Kernel Choice: Gaussian RBF

In this task, my kernel choice is the soft margin Gaussian Radial Basis Function (RBF) as it is a widely used kernel choice. Following the similar steps in previous SVM methods, we would first need to generate the gram matrix, which follows the equation below:

$$K(x_1, x_2) = \exp\left(-\gamma \|x_1, -x_2\|^2\right)$$

Equation 6. Gram matrix for RBF kernel

Like the soft margin nonlinear kernel, we will compute the discriminant function and parameters like Equation 5 and Figure 5. In this SVM, multiple values of γ and C will be used to generate the RBF SVM. The SVM that outputs the highest accuracy will be the final version of SVM_Main.

Classification Results

RBF Kernel	Train Accuracy (%)					Test Accuracy (%)				
	γ=0.001	γ=0.01	γ=0.1	γ=1	γ=10	γ=0.001	γ=0.01	γ=0.1	γ=1	γ=10
C=0.001	67.70	88.15	62.55	59.55	59.55	68.29	89.97	63.54	61.26	61.26
C=0.01	86.50	89	77.60	59.55	59.55	87.30	91.08	77.86	61.26	62.16
C=0.1	87.80	91.25	86.50	72.30	68.40	89.32	92.51	85.22	67.51	64.58
C=1	91.35	93.40	97.55	99.55	99.9	92.38	93.42	90.82	77.41	72.53
C=10	93.10	95.50	99.50	99.85	99.95	93.29	93.68	90.30	77.34	72.72
C=100	93..80	98.20	99.70	99.95	100	93.29	91.99	89.71	77.41	72.72

Table 2. Result of RBF Kernel SVM classification

By observing Table 2, we can see that when C has a low value of below 0.1, the accuracies of the SVM is mostly poor for both train and test accuracy. For $C \geq 1$, we can observe that training accuracy is all above 90%. However, at values where $\gamma \geq 1$, the test accuracy drops drastically.

Final γ and C parameter

As highlighted in red in table 2, the best results of the test accuracy would be chosen as the final parameters for SVM_Main:

$$\gamma = 0.01, C = 10$$