National University of Singapore

ME5404 Neural Network Part II

Project 1

SVM for Classification of Spam Email Messages

Name : Chua HongWei

Matriculation Number: A0074741E

Email: e0823141@u.nus.edu

25 Mar. 22

1. Introduction

Support vector machine (SVM) has proven to be a useful classification method that is robust to different input data. A decision boundary with large margin allows for more confidence classification of data where new data points having lesser influence to the decision boundary. By deploying soft margin on SVM controls the bias and variance trade-off, allowing some misclassification for overall prediction accuracies of the data. SVM is also proven to be useful for non-linearly separable data with kernel technique, further extend the application of SVM in the machine learning applications.

2. Project Task

This project explore the usage of SVM to classify spam dataset. The spam dataset contains a total of 4601 examples. Each containing a feature vector with 57 attributes that represent the selected key features of an email message, and a label indicating whether the associated email message is spam or not. More details of the dataset can be found in the project description sheet. In this project, we will explore the following three variations of SVM in this classification task.

1. Hard margin with linear kernel $K(x_1, x_2) = x_1^T x_2$
2. Hard margin with polynomial kernel $K(x_1, x_2) = (x_1^T x_2 + 1)^P$, with various p values
3. Soft margin with polynomial kernel $K(x_1, x_2) = (x_1^T x_2 + 1)^P$, with various p and C values

The label of the dataset are either "+1" for spam or "-1" for non-spam. Correctness of the classification will be judge based on accuracies of classification compared to the label.

3. Data Pre-processing

Data Visualization

As a start, we first explore the training and testing data that is provided in this project. The training set contains a total of 2000 data while the training set contains 1536 data.

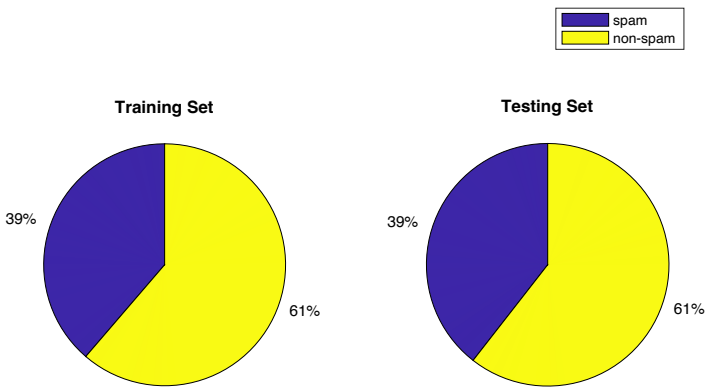

Figure 1 Distribution of training and testing set

The distribution of the data is shown on the pie chart in Figure 1. Both the training set and testing set are distributed similarly with the proportion of spam data to non-spam data equal for both dataset. This makes the classification task easier for the SVM model.

In this project, one self-initiatives effort done by me is some extra line of codes to check if the label are within the set of {-1, 1} with the 'find' function in Matlab. If there is a label which does not belongs to either -1 or 1, an error message will be displayed, warning of a possible data recording mistake in the datasets.

Data Standardization

As SVM is an classification model that involve distance between datapoints, it is recommended to perform data pre-processing technique on the datasets to prevent any outlier or feature vector with large magnitude from dominating the objective function. In this project, standardization method or mean removal and variance scaling is used to rescale the data to a Gaussian distribution with zero mean and unit variance. The data standardization is perform by calculating the standard score (z-score) of each data with the following formula

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

$$where\ \mu = mean\ of\ the\ feature, \sigma = standard\ deviation\ of\ the\ feature$$

Data standardization is performed in all the datasets including the training set, testing set and evaluation set with the mean and standard deviation calculated from the training set.

4. Mercer's Condition

In this project, two kernels, linear kernel and polynomial kernel with different p values was experimented for this SVM. Mercer's Condition was first used to determine the admissibility of the kernel before it is used in the classification task. The Mercer's condition is done by finding the Gram matrix first and followed by determining the eigenvalue of the Gram matrix. The kernel is non-admissible if any of the eigenvalue of the Gram matrix is less than zero. In the case of this project, a threshold of 1e-4 is used to represent an arbitrary small value. The Gram matrix formulation, K is illustrated below.

$$K = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \cdots & K(x_n, x_n) \end{bmatrix} where\ n = number\ of\ data$$

A self-created function named mercer.m was coded to calculate the Gram matrix, it's eigenvalue and output text of whether the kernel is admissible given the input parameters.

5. Kernels

In this project, three types of SVM kernel were explored to evaluate their respective performance in classifying the spam email in the given datasets.

Hard Margin with Linear Kernel

The linear kernel $K(x_1, x_2) = x_1^T x_2$ is a kernel where the hyperplane is a linear function. In 2-dimensional vector space, the linear function is a straight line, while in a 3-dimensional vector space, the linear function is a hyperplane. To solve the SVM optimization problem, we apply Lagrange function and KKT conditions on the SVM objective function to simplify the primal problem into a dual problem, where we only need to find the Lagrange multiplier α_i.

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

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

$$(2) \quad \alpha_i \geq 0$$

In Matlab, in order to perform the optimization and get the value of α_i, quadprog from the optimization toolbox is used with the following parameters.

$$\min_x \frac{1}{2} x^T H x + f^T x \quad \begin{cases} H(i,j) = d_i d_j x_i^T x_j \\ f = -\text{ones}(2000,1) \end{cases}$$

$$A \cdot x \leq b, \quad \begin{cases} A = [\,] \\ b = [\,] \end{cases}$$

$$Aeq \cdot x = beq, \quad \begin{cases} Aeq = train\_label' \\ beq = 0 \end{cases}$$

$$lb \leq x \leq ub. \quad \begin{cases} lb = zeros(2000,1) \\ ub = \text{ones}(2000,1) * C \end{cases}$$

$$x0 = [\,] \qquad options = optimset('LargeScale','off','MaxIter',1000)$$

In theory, for hard margin, the C value is infinity large. For the value of C during the implementation of SVM in Matlab, an arbitrary large number of 10e6 is used. After α_i is obtained, we will be calculate the optimal weight, w_o and b_o with the following method.

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

where $\mathbf{x}^{(s)}$ is a support vector with label $d^{(s)}$

For hard margin, b_o is calculated by selecting one of the support vector and it's corresponding label.

The discriminant function g(x) is then calculated with the following formula.

$$g(\mathbf{x}) = \mathbf{w}_o^T \mathbf{x} + b_o$$

A function named discriminant_func.m is coded to perform this task.

Hard and Soft Margin with Polynomial Kernel

The non-linear kernel used in this project is the polynomial kernel $K(x_1, x_2) = (x_1^T x_2 + 1)^P$. P value is varied in this project from 1 to 5. The objective function and constraints for polynomial kernel are as follows.

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

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

Similar to linear kernel, quadprog with the following parameters is used to solve the optimization problem.

$$\min_x \frac{1}{2} x^T H x + f^T x \quad \begin{cases} H(i,j) = d_i d_j (x_1^T x_2 + 1)^p \\ f = -\text{ones}(2000,1) \end{cases}$$

$$A \cdot x \le b, \quad \begin{cases} A = [\,] \\ b = [\,] \end{cases}$$

$$Aeq \cdot x = beq, \quad \begin{cases} Aeq = train\_label' \\ beq = 0 \end{cases}$$

$$lb \le x \le ub. \quad \begin{cases} lb = zeros(2000,1) \\ ub = \text{ones}(2000,1) * C \end{cases}$$

$x0 = [\,]$ $\quad options = optimset('LargeScale','off','MaxIter',1000)$

For soft margin task, the penalty term C, is varied from 0.1, 0.6, 1.1, 2.1 to find the effect of different C value on SVM classification accuracies. After α_i is obtained, w_o and b_o were calculated with the following method.

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} \, d_i \, \mathbf{x}_i \qquad b_{o,i} = \frac{1}{d_i} - \mathbf{w}_o^T \mathbf{x}_i \qquad b_o = \frac{\sum_{i=1}^{m} b_{o,i}}{m}$$

The difference between the calculation of b_o for non-linear kernel when compare to linear kernel is that the b_o is calculated by finding the average of all the b that are support vectors.

The discriminant function g(x) is then calculated with the following formula.

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

After the g(x) is obtained, the SVM can be used for classification task of new data with the following formula

$$d_{new} = sgn(g(X_{new}))$$

6. Evaluation Metric

The evaluation metric used in this project is accuracy. We calculate the fraction of correct prediction over the total number of data. Accuracy could be represented with the following formula

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

The accuracy score is converted into percentage for this project.

7. Results

Table 1 Results of SVM Classification

Type of SVM	**Training accuracy**				**Test accuracy**			
Hard margin with Linear kernel	73.90%				75.33%			
Hard margin with Polynomial kernel	**p = 2**	**p = 3**	**p = 4**	**p = 5**	**p = 2**	**p = 3**	**p = 4**	**p = 5**
	99.90%	100%	100%	NA	86.72%	86.78%	84.11%	NA
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	93.00%	93.25%	93.35%	93.35%	93.62%	93.49%	93.68%	93.68%
p = 2	98.70%	99.20%	99.35%	99.50%	90.23%	88.41%	88.02%	88.02%
p = 3	99.65%	99.75%	99.85%	99.85%	90.43%	89.91%	89.45%	89.13%
p = 4	99.80%	99.90%	99.90%	99.90%	88.82%	87.76%	87.37%	87.24%
p = 5	NA	NA	NA	NA	NA	NA	NA	NA

NA = Not admissible

NA in the table indicate kernel which fails Mercer's condition and deem not a suitable kernel to use for SVM classification. From the table, the combination which produced the best test accuracy for each category (indicated in green) is as follows.

1. Hard margin with linear kernel – 75.33%
2. Hard margin with polynomial kernel (p = 3) – 86.78%
3. Soft margin with polynomial kernel ([p = 1, C = 1.1], [p = 1, C = 2.1]) – 93.68%

8. Result Discussion

Hard Margin with Linear Kernel

The hard margin with linear kernel produced an accuracy of 73.90% for training set, indicating that the data is not linearly separable. I further tested the algorithm by choosing different support vector randomly to compute the b_0 value. It is found that the b_0 value varies for different support vector, further proven that the data is not linearly separable and linear kernel is not suitable for this task.

Hard Margin with Polynomial Kernel

Hard margin with polynomial kernel produced a good training accuracy for p = 2, 3 and 4, with accuracy of close to 100%. This suggest that the dataset is linearly separable at high dimensional feature space. However, it is observe that overall test accuracies decreases to 84% – 86% for p = 2,3 and 4 which indicates that overfitting occurs. For p = 5, the Mercer's condition is not met and is not a suitable kernel for this dataset.

Soft Margin with Polynomial Kernel

For soft margin with polynomial kernel, it can be seen that when p = 5, the kernel fail to satisfy the Mercer's condition and thus not suitable kernel to use for this dataset. From table 1, the training accuracy for this method is high at more than 90% for different penalty term, C, indicating the suitability of non-linear kernel on this dataset.

As for the testing data, polynomial kernel with p = 1 produces the best accuracy score. When compare the test accuracy and training accuracy for polynomial kernel with p = 1, it is observed that both training and test accuracy didn't differ too much, indicating that this kernel is a proper fit to both the training and testing data. In fact, polynomial kernel with p = 1 and C = 1.1 and 2.1 produced the best test accuracy among the different soft margin with polynomial kernel variations.

Another trend observed from the experiment is that the test accuracy decreases as the C value increases for p = 2, 3 and 4. This is probably due to smaller margin lead by high C values that is less tolerate to misclassification.

9. SVM Design (Task 3)

For this task, I had created an extra Radial Basis Function (RBF) kernel as it is a common kernel to go for non-linearly separable dataset due to its similarity to Gaussian Distribution. The RBF kernel can be mathematically represented as follows.

$$K(x_1, x_2) = \exp(-\frac{\|x_1 - x_2\|^2}{2\sigma^2})$$

Sigma σ, is the hyperparameters representing how far is the influence of each training point, with large σ representing 'far' and small σ represents 'close'. Soft margin is apply together with the RBF kernel.

Table 2 Results of RBF SVM Classification

Type of SVM	**Training accuracy**				**Test accuracy**			
Soft margin with RBF 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**
sigma = 0.1	62.25%	99.95%	99.95%	99.95%	60.94%	65.89%	66.41%	66.47%
sigma = 1	89.40%	97.40%	98.80%	99.55%	88.15%	92.58%	93.23%	93.29%
sigma = 2	92.30%	94.90%	96.00%	97.85%	91.93%	94.08%	94.34%	94.34%
sigma = 3	91.55%	94.10%	94.70%	95.65%	91.67%	94.01%	94.34%	94.73%
sigma = 4	90.95%	93.35%	94.30%	94.80%	91.08%	93.23%	94.34%	94.53%

The soft margin with RBF kernel produce better performance when compare to hard margin with linear kernel, hard margin with polynomial kernel and soft margin with polynomial kernel with highest accuracy of 94.73% achieved with parameters sigma = 3 and C = 2.1. Thus, I had selected this model for the evaluation data.

A Matlab code, dummy_data.m,was coded to randomly generate 600 dummy data from the combination of training data and testing data. Several trial were run on svm_main.m and the results are as follows.

Trial	1	2	3	4
Accuracy	93.33%	95.33%	96.33%	95.17%

The result show robustness of the kernel in this classification task, with the accuracy consistently above 93%.

10. Code File

EDA.m – Exploratory Data Analysis of the dataset.

svm_task1and2.m – Matlab file for task 1 & 2.

svm_main – Matlab file for task 3.

mercer.m – Function file to check for Mercer's condition.

discriminant_func.m – Function file to get the discriminant function, g(x).

dummy_eval.m – Matlab file to generate eval_data randomly.