Project 1: SVM for Classification of Spam Email Messages

Name :	LUO ZIJIAN
Matric.No：	A0224725H
Email:	luozijian@u.nus.edu
Module：	NEURAL NETWORKS(EE5904)

Task 1:

The goal of task 1 is to record the performance of different type of SVM rule in training data. And compare the different impact of different parameters.

Here are three types of SVM rule.

(1) Hard-margin SVM with the linear kernel

$$K(x_1, x_2) = X_1^T X_2$$

(2) Hard-margin SVM with a polynomial kernel

$$K(x_1, x_2) = (X_1^T X_2 + 1)^p$$

(3) Soft-margin SVM with a polynomial kernel

$$K(x_1, x_2) = (X_1^T X_2 + 1)^p$$

Followed by the instruction from GA, our algorithm is justified by the accuracy, so different SVM with different parameters can get different result. But in practice, some special case would not satisfy some conditions. For example, The Gram matrix's eigenvalues are non-negative in theory. However, in practice, we should choose an appropriate threshold for the eigenvalues because of the limitation of modern computers. In this task, we should set this threshold for eigenvalues 10^{-4}. If any eigenvalue of gram matrix is smaller than the threshold, this kernel candidate is not admissible.

Total process:

First step is to pre-processing data, in this project, I use the standard normalized training data to calculate. Second step is to use 'quadprog' function to get separate alpha of different SVM. Thirdly, we use this alpha to calculate related discrimination function to record SVM performance. Last step is to calculate the training accuracy.

Task 2:

The goal of task 1 is to record the performance of different type of SVM rule in testing data. Based on our record in task1, we can calculate the testing accuracy.

Total process:

The total process is similar as task1, but in this part, we need to use testing data. And combined the result from task1, we can get this table below.

Table.1 Result of SVM classification

Type of SVM	Training accuracy				Test accuracy			
Hard margin with Linear kernel	93.50%				93.23%			
Hard margin with Polynomial kernel	P=2	P=3	P=4	P=5	P=2	P=3	P=4	P=5
	98.90%	99.30%	Non-convex	Non-convex	85.87%	85.93%	Non-convex	Non-convex
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	98.80%	99.50%	99.50%	99.55%	89.38%	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%	100.00%	100.00%	87.89%	86.78%	86.00%	86.00%
P=5	98.90%	98.80%	98.95%	99.15%	87.30%	86.39%	86.00%	86.13%

From result, we can get these conclusions:

1. For hard-margin with linear kernel, the training performance is worse than that of polynomial kernel, because with 99.99% versus 93.50%, except when p=4 or 5. But when it comes to the testing result, the performance is worse than linear kernel.
2. For the same polynomial kernel. The performance of soft-margin is better than that of hard-margin. When p equals 4 or 5, both soft-margin and hard-margin could not satisfy the requirement of Mercer's condition, the quadprog function can still work well because the soft margin guarantee it is still convex when C is small.
3. Especially for soft margin, when p=4, the accuracy can get 100%. But when we focus on the test result, the testing accuracy is reverse condition. I suppose that the training process is overfitting, so this parameter is not reliable.
4. As for the non-convex condition in hard margin with polynomial kernel, I found this case was caused by the negative eigenvalues, which apply this problem appears non-convex, so it could not reach the minimum and this is unbounded. Therefore, in this condition, it is not admissible.

In conclusion, I suppose that hard margin with linear kernel or soft margin with low p value can get a higher accuracy for testing data. For other conditions, they may be higher accuracy in training result.

Task 3

In order to design a SVM of better performance, I make some tests with different parameters. Based on the result from task1 and task2, we know when p is greater, the SVM result may be overfitting

1. **The type of kernel**
   I just choose the soft margin with polynomial kernel as my SVM kernel. Based on the better performance both in training data and testing result.
2. **The effect of C (The threshold for SVM), when p=1;**

Value of C	Train accuracy	Test accuracy	Total accuracy
0.1	92.65%	92.97%	92.79%
0.2	92.65%	92.90%	92.76%
0.3	92.70%	92.90%	92.79%
0.4	92.55%	92.70%	92.62%
0.5	92.50%	92.90%	92.68%
1.0	92.65%	93.03%	92.82%
2.0	92.90%	93.22%	93.04%
3.0	92.85%	93.16%	92.99%
4.0	92.80%	93.16%	92.96%
5.0	93.10%	93.23%	93.16%
10.0	93.30%	93.16%	93.24%
100.0	93.55%	93.16%	93.38%
1000.0	93.40%	93.16%	93.38%
10000.0	93.50%	93.23%	93.38%
100000.0	93.50%	93.23%	93.38%

Through the result, I found the total accuracy get the limitation of 93.38% when C is big enough. So, there is threshold in C, that is around 100.

So, I detect the performance of C around 100.

Value of C	Train accuracy	Test accuracy	Total accuracy
70.0	93.35%	93.16%	93.27%
80.0	93.35%	93.23%	93.30%
90.0	93.40%	93.23%	93.33%
100.0	93.55%	93.16%	93.38%
110.0	93.60%	93.16%	93.41%
120.0	93.60%	93.10%	93.38%
130.0	93.60%	93.10%	93.38%
140.0	93.60%	92.96%	93.33%

150.0	93.60%	92.97%	93.33%

Through the result, I found the total accuracy get the optimal result of 93.41% when C is 110.

From all the results, we can get the highest accuracy when C is 110.
So, I set this value in task3.

Coding remarks:

Each SVM in Task1 and task2 are combined in a single SVM code, if you want to run this Matlab file, please put the train.data and test.data into the same working space with these files.

As for task2, this file has been saved in svm_main.m. If you want to run this Matlab file, please put eval.data and train.data into same working space with this file.