SVM FOR CLASSIFICATION OF SPAM EMAIL MESSAGES

MODULE:EE5904 NEURAL NETWORKS

NAME: VENKATARAMAN MEENAKSHI

MATRIC ID: A0247125J

EMAIL ID: e0920432@u.nus.edu

Task 1) Computing the discriminant function for each of the case as follows:

i) Hard Margin with Linear Kernel

In order to obtain the discriminant function the following steps have been followed:

Data Preprocessing:

The train data has been standardized using the mean and standard deviation of each feature used to represent the input data. Thus, this preprocessed data is obtained as follows:

$$\text{Preprocessed Input} = \frac{\text{Input} - \text{Mean}}{\text{Standard Deviation}}$$

Using the mean and standard deviation of the training data, the standardized test data is also computed and used.

Checking for admissibility of Kernel using Mercer's Condition:

Kernel used $K = x1^{T}x2$

Mercer's condition is used to check if convergence of solution is guaranteed when the given kernel is used. When this condition fails, although optimal solution can be obtained for certain dimensions of data, this will not observed for all types of data that are utilized. In order to check for Mercer's Condition, the eigen values of the linear kernel that has been computed using the preprocessed input data are used to check for admissibility of the kernel wherein if any eigen value is found to be less than 0, the kernel is not considered to be admissible. Since eigen values are found to be very small in value, the absolute value 0 cannot be used and thus a threshold of -1e-6 has been set as the reference value instead of zero for checking Mercer's condition.

```
Kernel admissible
```

The above output is obtained, thereby indicating that the kernel satisfies Mercer's Condition and is therefore admissible.

Finding the Optimal Hyperplane using method of Lagrange Multiplier:

Given the input x_i and the desired output d_i, the lagrange multipliers α_i need to be found such that

$Q(\alpha)=\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j x_i^T x_j$ is maximized. Here $x_i^T x_j$ represents the linear kernel.

Since we need to find a minimal solution, the equivalent maximum will be - Q(α) and thus becomes

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

Thus in order to minimize as given above, the following parameters have been provided as an input to the Quadprog function in order to find an optimal solution.

```
H=zeros(size(K_linear,1),size(K_linear,1));
for i=1:size(K_linear,1)
    for j=1:size(K_linear,1)
        H(i,j)=y_train(i)*y_train(j)*K_linear(i,j);
    end
end
options=optimset('LargeScale','off','MaxIter',2000);
f=-1*ones(1,size(K_linear,1));
A=[];
b=[];
Aeq=transpose(y_train);
beq=0;
lb=zeros(size(K_linear,1),1);
C=10^6;
ub=C*ones(size(K_linear,1),1);
x0=[];
alpha=quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options);
```

When the above function quadprog function is implemented, it is observed that all solutions are convex and thus optimal solutions are thus found. However convergence of the solution is dependent on the number of maximum iterations that are being set. The function as shown above makes use of 2000 iterations to reach an optimal solution since C is set to be a very large value. This value of C has been set as 10^6 in order to depict the hard margin constraint.

Once the alpha values have been found, they are in turn used to obtain the weight and bias values.

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

Once the optimal solution has been found, the support vectors are found using the threshold 1e-4 which has been used in place of the ideal value of 0.
Using these support vectors, the the bias for the discriminant function is calculated.

$$b_o = \frac{1}{d^{(s)}} - \mathbf{w}_o^T \mathbf{x}^{(s)}$$

The bias for the discriminant values is then taken as the mean of the bias values that are obtained for each support vector.

Thus the discriminant function is then considered to be

$g(x) = w_o^T x + b_o$

```
K_linear_test=zeros(size(x_train,1),size(x_test,1));
for i=1:size(K_linear_test,1)
    for j=1:size(K_linear_test,2)
        K_linear_test(i,j)=x_train_preprocessed(i,:)*transpose(x_test_preprocessed(j,:));
    end
end

g_test=zeros(size(x_test,1),1);
for i=1:length(x_test)
    for j=1:size(K_linear_test,1)
        g_test(i)=g_test(i)+((alpha(j)*y_train(j)*K_linear_test(j,i)));

    end
    g_test(i)=g_test(i)+b0_hm_linear;
end
```

ii) Hard Margin with Polynomial Kernel:

In the case of hard margin with polynomial kernel, the standardized data is used to check for Mercer's Condition. The nonlinear kernels are represented as:

$K=(x1^T x2+1)^p$

Polynomial Degree	2	3	4	5
Admissibility	Admissible	Admissible	Not Admissible	Not Admissible

In the case of non linear kernel,-Q(α)=$-\sum_{i=1}^{N}\alpha_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j (1 + x_i^T x_j)^p$ is maximized, i.e Q(α) is minimized. The following parameters have been provided to the quadprog function in order to solve for optimization:

```
K_nonlinear=(x_train_preprocessed*transpose(x_train_preprocessed) + 1).^p(m);

H=zeros(size(K_nonlinear,1),size(K_nonlinear,1));
for i=1:size(K_nonlinear,1)
    for j=1:size(K_nonlinear,1)
        H(i,j)=y_train(i)*y_train(j)*K_nonlinear(i,j);
    end
end
options=optimset('LargeScale','off','MaxIter',1000);
f=-1*ones(1,size(K_nonlinear,1));
A=[];
b=[];
Aeq=transpose(y_train);
beq=0;
lb=zeros(size(K_nonlinear,1),1);
C=10^6;
ub=C*ones(size(K_nonlinear,1),1);
x0=[];
alpha=quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options);
```

The formulated problem is found to be convex and therefore optimal solutions are found that are found to satisfy the provided constraints.

The discriminant function is then calculated as follows:

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

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

Here m refers to the number of support vectors obtained while N refers to the total number of training samples present.

```
K_nonlinear_test=zeros(size(x_test,1),size(x_train,1));
for i=1:size(K_nonlinear_test,1)
    for j=1:size(K_nonlinear_test,2)
        K_nonlinear_test(i,j)=((x_test_preprocessed(i,:)*transpose(x_train_preprocessed(j,:)))+1).^p(m);
    end
end

g_test=zeros(size(x_test,1),1);
for i=1:length(x_test)
    for j=1:size(K_nonlinear_test,2)
        g_test(i)=g_test(i)+((alpha(j)*y_train(j)*K_nonlinear_test(i,j)));

    end
    g_test(i)=g_test(i)+b0_hm_nonlinear;
end
```

iii) Soft Margin with Non Linear Kernel:

In the case of usage of non linear kernels with soft margin, the value of C is reduced to a smaller value as compared to that which is used for hard margins, so as to allow certain amount of misclassification. The amount of misclassification that is allowed varies with the value of C which is set. The allowed number of misclassifications increases as the value of C decreases.
The support vectors in this case are considered to be alpha values that are found to be greater than or equal to 0 and lesser than or equal to the C value that is utilized.

Checking for Mercer's Condition using standardized data:

	C=0.1	C=0.6	C=1.1	C=2.1
P=1	Admissible	Admissible	Admissible	Admissible
P=2	Admissible	Admissible	Admissible	Admissible
P=3	Admissible	Admissible	Admissible	Admissible
P=4	Not Admissible	Not Admissible	Not Admissible	Not Admissible
P=5	Not Admissible	Not Admissible	Not Admissible	Not Admissible

From the above table, it is observed that nonlinear kernels making use of 4 and 5 degrees of polynomials do not satisfy Mercer's Condition.

The discriminant function obtained follows the same function as that obtained for polynomial kernel using hard margin, the only difference being the range defined for selection of support vectors.

Task 2:

Implementing the trained SVM Classifiers on the train and test data that have been provided so as to compare the performance of results obtained from training using different values of the penalty C and the degree of polynomial p:

Type of SVM	**Train Accuracy**				**Test Accuracy**			
Hard Margin with Linear Kernel	93.55				93.82			
Hard Margin with Non linear Kernel	**p=2**	**p=3**	**p=4**	**p=5**	**p=2**	**p=3**	**p=4**	**p=5**
	99.80%	100.00%	Not Admissible	Not Admissible	86.20%	86.78%	Not Admissible	Not Admissible
Soft Margin with Non linear 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	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	Not Admissible	Not Admissible	Not Admissible	Not Admissible	Not Admissible	Not Admissible	Not Admissible	Not Admissible
p=5	Not Admissible	Not Admissible	Not Admissible	Not Admissible	Not Admissible	Not Admissible	Not Admissible	Not Admissible

Discussion of Results in table:

As can be seen from the table above, it is observed that Hard Margin as well as Soft Margin with Linear Kernel(p=1) are found to produce the best results in terms of test accuracy while Hard Margin as well as Soft Margins used in Non Linear Kernels provide higher train accuracy results. Soft Margin based Non linear kernel having degree of polynomial as 3 is found to produce better results as compared to that using degree of polynomial 2. It can also be seen that for non linear kernels using soft margin, the test accuracy results are found to be better for lower values of C.

Although hard margin based kernels in this case yield good results, in general, the usage of hard margin can lead to overfitting of the samples provided and therefore will not be able to classify data in the expected manner if the data cannot be distinctly separated. The usage of higher degree of polynomials on the other hand can result in increased complexity which can therefore contribute to more overfitting and thus classifiers will not be able to generalize the classification of data.

In the case of polynomial degrees 4 and 5 respectively, the kernels using hard margin as well as soft margin are found to have negative eigenvalues and thus do not satisfy Mercer's Condition as a result of which their corresponding classifier performances are not being considered. However, this does not imply that optimal hyperplane cannot be obtained.

Task 3) Training SVM Classifier using own kernel:

The kernel used is the Laplacian kernel that is defined as:

K= $e^{(-\gamma|x_i - x_j|)}$ where $|x_i - x_j| = \sum |x_i - x_j|$

The given kernel is generated using the input training data as per the function defined following which it is checked using Mercer's Condition in order to determine its admissibility. It is observed that all eigenvalues are non negative and thus the given kernel is admissible. The hyperparameters of the classifier which include the C value as well as γ which is used in the kernel function are tuned in order to find the optimal values.

In order to find the optimal C and gamma values, the performance of the SVM classifier using different values of C and gamma, is compared so as to observe and determine the C and gamma parameters that are to be used. The following table shows the train and test accuracies that have been obtained using this kernel function through predictions on sample data values.

C	Gamma	Test Accuracy	Train Accuracy
10	0.1	83%	100%
5	0.1	82%	99%
10	0.05	82%	99%
50	0.1	82%	100%
100	0.01	82%	99%
100	0.1	82%	100%
50	0.01	82%	99%
50	0.05	82%	100%
100	0.005	82%	99%
5	0.05	82%	99%
50	0.005	82%	98%
100	0.05	82%	100%
1	0.1	81%	98%
0.6	0.1	81%	97%
10	0.01	81%	97%
5	0.01	80%	96%
10	0.005	80%	96%
1	0.05	80%	96%

The above table provides the different accuracy values obtained from varying the C and Gamma values in the order of decreasing test accuracies. It can be seen that high test accuracy results are obtained for C values equal to 5,10,50 and 100 and for gamma values equal to 0.05,0.01 and 0.1 respectively as highlighted in dark blue above. The train accuracies for these results are found to be extremely high and thus indicate overfitting. The values are thus chosen from the lower end of this table so as increase generalization of the classifier. Since there is not much difference in the values of the test accuracy, the C value

is chosen such that it does not allow too many misclassifications as well as does not impose large amount of penalty as well. The C value is thus chosen to be 10 while the value of gamma corresponding to this value in the table is observed to be 0.01. The train and test accuracy values are given in the table below:

Train Accuracy	Test Accuracy
0.967	0.953776042

The following figure shows the implementation of the discriminant function using the kernel function that is used.

```
K_nonlinear_test=zeros(size(x_train,1),size(x_test_preprocessed,1));
for i=1:size(K_nonlinear_test,1)
    for j=1:size(K_nonlinear_test,2)
        %K_nonlinear_test(i,j)=((x_train_preprocessed(i,:)*transpose(x_test_preprocessed(j,:)))+1).^p(m);
        K_nonlinear_test(i,j)=exp(-0.01*sum(abs(x_train_preprocessed(i,:)-x_test_preprocessed(j,:))));
        %K_nonlinear_test(i,j)=(1/2)*(tanh((x_train_preprocessed(i,:)-2)./2)*transpose(tanh((x_test_prepro
    end
end

g_test=zeros(size(x_test,1),1);
for i=1:length(x_test)
    for j=1:size(K_nonlinear_test,1)
        g_test(i)=g_test(i)+((alpha(j)*y_train(j)*K_nonlinear_test(j,i)));

    end
    g_test(i)=g_test(i)+b0_hm_nonlinear;
end
```