CMPE 462 – PROJECT 2

Implementing an SVM Classifier

- **Student ID1:** 2014400051
- **Student ID2:** 2018400291
- **Student ID3:** 2019705072

Requirements and How to Run:

- The project uses some python libraries that need to be installed before starting the project. Namely, scipy.io, numpy, libsvm, prettytable.
- Also, the data needs to be in the folder data under data which is in the same workspace as the .ipynb file.

If you would like to load data from another folder. Since the data is matlab file, need to use loadmat function.

Specify it in the data loading cell. Dot means current working directory.

```
!pip3 install libsvm
!pip3 install PTable
```

```
Collecting libsvm
Installing collected packages: libsvm
Successfully installed libsvm-3.23.0.4
Collecting PTable
Installing collected packages: PTable
Successfully installed PTable-0.9.2
```

```
import scipy.io as spio
from libsvm.svmutil import *
import numpy as np
from prettytable import PrettyTable

dic = spio.loadmat("data.mat")

train_matrix, train_labels = dic['X'][:150], dic['Y'][:150]
test_matrix, test_labels = dic['X'][150:], dic['Y'][150:]

train_labels = train_labels.reshape((train_labels.shape[0],))
test_labels = test_labels.reshape((test_labels.shape[0],))
```

INTRODUCTION:

In this project, we implemented support vector machines. We start with hard margin linear SVM and continue with soft margin SVM with different C values and different kernel functions, respectively. After these implementations, we tried to observe the theory in application, in a given dataset. Finally we investigate the changes in hyperplane when we remove one support vectors and one data point that is not a support vector.

TASK 1:

We implemented the hard margin linear SVM thanks to the software package LIBSVM, and have got the following train and test classification accuracies for the given data. In order to get hard-margin SVM, we choose large C which is 100000000 to force error tolerance to be 0. When we increase the value of C further after it is 100000000, Train and test classification accuracy do not change. Thus, we get hard-margin SVM.

```
Hard Margin SVM
Train classification accuracy: 74.6667
Test classification accuracy: 77.5
```

Test classification accuracy is higher than training classification accuracy. The reason behind it might be that data points in training dataset may be hard to decide classify and in test dataset may be easy to decide classify.

TASK 2:

In this task we want to investigate soft margin SVM and what is happening when we keep kernel fixed, and try different C-values. Also, when we keep fixed C-values, and try different kernel functions.

Part 1: Fixed Kernel Functions, Different C-Values

In part 1, we try different fixed kernel functions and different C-values for each fixed kernel functions.

a) Kernel function: Radial Basis Function

Kernel -> Radial basis function : exp(-1*|u-v|^2)								
C Value	0.001	0.01	0.1	1	10	100	1000	10000
Train Accuracy	53.3333	53.3333	83.3333	86.6667	95.3333	99.3333	100.0	100.0
Test Accuracy	58.3333	58.3333	84.1667	84.1667	77.5	78.3333	76.6667	76.6667

For the kernel, radial basis function, we observe that when C value increases, train accuracies increase and test accuracies can fluctuate. For C values between 100 and 1000, model reaches the 100% training accuracy and 76.7% test accuracy, keeps constant for increased C values that we tried. When we increase the value of C further after it is 10000, training and test accuracy do not change. Thus, we can easily say that value of C at 1000, we get hard-margin SVM.

b) Kernel function: Quadratic Polynomial Function

Kernel -> Polynomial: (u'*v)^2								
C Value	0.001	0.01	0.1	1	10	100	1000	10000
Train Accuracy	53.3333	53.3333	56.0	85.3333	88.6667	97.3333	99.3333	100.0
Test Accuracy	58.3333	58.3333	59.1667	78.3333	79.1667	76.6667	73.3333	69.1667

For the kernel, quadratic polynomial, function, we observe that when C value increases, train accuracies increase and test accuracies can fluctuate. For C values between 1000 and 10000, model reaches the 100% training accuracy and 69.17% test accuracy, keeps constant for increased C values that we tried. When we increase the value of C further after it is 10000, training and test accuracy do not change. Thus, we can easily say that value of C at 10000, we get hard-margin SVM.

c) Kernel function: Sigmoid Tanh Function

Kernel -> Sigmoid: tanh(u'*v)								
C Value	0.001	0.01	0.1	1	10	100	1000	10000
Train Accuracy	53.3333	53.3333	82.0	82.6667	78.0	76.6667	75.3333	75.3333
Test Accuracy	58.3333	58.3333	84.1667	84.1667	80.0	72.5	74.1667	74.1667

For the kernel, sigmoid tanh, function, we observe that when C value increases, train accuracy increases while C value reaches to a value between 0.1 and 1, with that value train accuracy has suddenly decreased. Then, it follows the decreasing trend. Also, same observation is valid for test accuracy, too. Finally, model reaches the 75.3% training accuracy and 74.17% test accuracy at C-value somewhere between 100 and 1000. When we increase the value of C further after it is 10000, training and test accuracy changed. Thus, we can easily say that we get hard-margin SVM at C is greater than 10000.

d) Kernel function: Linear Function

Kernel -> Linear: u'*v								
C Value	0.001	0.01	0.1	1	10	100	1000	10000
Train Accuracy	53.3333	82.6667	86.0	86.6667	88.6667	88.6667	90.0	90.0
Test Accuracy	58.3333	84.1667	83.3333	85.0	81.6667	81.6667	81.6667	81.6667

For the kernel, linear, function we observe that when C value increases, train accuracies increase and test accuracies both increase and decrease. For C values between 1 and 10, model reaches the 88.67% training accuracy and it improves for the higher C-values, but test accuracy has pick at C value =1, with 85% then it decreases to 81.67% and stays there for desired C-values. When we train model with C = 100000000 , We get hard-margin SVM and When we increase the value of C further after it is 100000000 , training and test accuracy do not change.

e) Kernel function: Cubic Polynomial Function

Kernel -> Polynomial: (u'*v)^3								
C Value	0.001	0.01	0.1	1	10	100	1000	10000
Train Accuracy	53.3333	53.3333	53.3333	86.0	94.0	98.6667	100.0	100.0
Test Accuracy	58.3333	58.3333	58.3333	82.5	80.8333	75.0	75.8333	75.8333

For the kernel, cubic polynomial, function we observe that when C value increases, train accuracy increases, but test accuracies can fluctuate. For C values between 100 and 1000, model reaches the 100% training accuracy and 75.83% test accuracy, keeps constant for increased C values that we tried. When we increase the value of C further after it is 1000, training and test accuracy do not change. Thus, we can easily say that value of C at 1000, We get hard-margin SVM.

Part 2: Fixed C-Values, Different Kernel Functions

In part 2, we try different fixed C-Values and different kernel functions for each fixed C-Values.

a) Fixed C = 0.01

Fixed C = 0.01					
Kernel	polynomial(U'V)^3	polynomial(U'V)^2	linear	sigmoid	RBF
Train Accuracy	53.3333	53.3333	82.6667	53.3333	53.3333
Test Accuracy	58.3333	58.3333	84.1667	58.3333	58.3333

For the fixed C= 0.01, the both highest train and test accuracy is given by the linear kernel function among the 5 different kernel functions shown in the above table.

b) Fixed C = 0.1

Fixed C = 0.1					
Kernel	polynomial(U'V)^3	polynomial(U'V)^2	linear	sigmoid	RBF
Train Accuracy	53.3333	56.0	86.0	82.0	83.3333
Test Accuracy	58.3333	59.1667	83.3333	84.1667	84.1667

For the fixed C= 0.1, the highest train accuracy is given by the linear kernel function, and highest test accuracy is given by the both sigmoid and RBF kernel function among the 5 different kernel functions shown in the above table.

c) Fixed C = 1

Fixed C = 1					
Kernel	polynomial(U'V)^3	polynomial(U'V)^2	linear	sigmoid	RBF
Train Accuracy	86.0	85.3333	86.6667	82.6667	86.6667
Test Accuracy	82.5	78.3333	85.0	84.1667	84.1667

For the fixed C=1, the highest train accuracy is given by the both linear and RBF kernel function, and highest test accuracy is given by the linear kernel function among the 5 different kernel functions shown in the above table.

d) Fixed C = 10

Fixed C = 10					
Kernel	polynomial(U'V)^3	polynomial(U'V)^2	linear	sigmoid	RBF
Train Accuracy	94.0	88.6667	88.6667	78.0	95.3333
Test Accuracy	80.8333	79.1667	81.6667	80.0	77.5

For the fixed C=10, the highest train accuracy is given by the RBF kernel function, and highest test accuracy is given by the kernel linear function among the 5 different kernel functions shown in the above table.

e) Fixed C = 100

Fixed C = 100					
Kernel	polynomial(U'V)^3	polynomial(U'V)^2	linear	sigmoid	RBF
Train Accuracy	98.6667	97.3333	88.6667	76.6667	99.3333
Test Accuracy	75.0	76.6667	81.6667	72.5	78.3333

For the fixed C=100, the highest train accuracy is given by the RBF kernel function, and highest test accuracy is given by the linear kernel function among the 5 different kernel functions shown in the above table.

f) Fixed C = 1000

Fixed C = 1000					
Kernel	polynomial(U'V)^3	polynomial(U'V)^2	linear	sigmoid	RBF
Train Accuracy	100.0	99.3333	90.0	75.3333	100.0
Test Accuracy	75.8333	73.3333	81.6667	74.1667	76.6667

For the fixed C=1000, the highest train accuracy is given by the both cubic and RBF kernel function, and highest test accuracy is given by the kernel linear function among the 5 different kernel functions shown in the above table.

All in all, in all fixed C values, generally linear kernel gives highest test accuracy and RBF kernel gives highest train accuracy.

TASK 3:

In this task we want to investigate the relationship between number of support vectors and C-Value by remaining all other parameters the same.

For this investigation, we used the linear function as a kernel, started with the C Value 0.001 and increased 10 times for each iteration. As a result, the following table appeared.

Kernel -> Linear: u'*v								
C Value	0.001	0.01	0.1	1	10	100	1000	10000
Number of Support Vectors	140	118	74	58	51	50	49	49

C is penalty term on the slack variables, measures the degree to which the margin constraints are violated. If we increase C-value, a greater penalty is applied on violators and less data points are tolerated. So, SVM margin will be narrower. Thus, less data points are support vectors. Theory says when we increase value of C, number of support vectors decreases. Our observations match the theory.

TASK 4:

In this task we want to investigate the relationship between the hyperplane and support vectors and a data point that is not a support vector.

For this purpose, we used linear kernel function and chose 10 as C value. In every move, we trained the model and observed the weight, norm of the weight and number of support vectors.

Our weights, norm of the weight and number of support vectors are like the following output just before not removing any points.

```
Weight: [-1.30726513  0.52514823  1.34794553  1.38300601  1.61374734  0.15936149
  0.13144559 -1.8510571  -0.0279832  -0.07509238  0.07069583  2.72990177
  0.44450478]
Norm of the weight: 4.4101043605317525
Number of Support Vectors: 51
```

Removing one data point that is not a support vector

```
Weight: [-1.30726513  0.52514823  1.34794553  1.38300601  1.61374734  0.15936149
  0.13144559 -1.8510571  -0.0279832  -0.07509238  0.07069583  2.72990177
  0.44450478]
Norm of the weight: 4.410104360531701
Number of Support Vectors: 51
```

Nothing has changed when we removed one data point that is not a support vector. This result should be expected, because it is not a move to change the hyperplane or the margin.

Removing one of the support vectors

```
Weight: [-1.35468538  0.52540875  1.34077722  1.38151342  1.63622204  0.22703252
  0.16792452 -1.97393535 -0.00791992 -0.16670834  0.01305132  2.50296982
  0.51196652]
Norm of the weight: 4.3618651742011005
Number of Support Vectors: 53
```

In order to get accurate result, we tried all possibilities. We chose one of the 51 support vectors and did it 51 times to get accurate result. Statistics are given below.

```
There are 51 support vectors.
36 times norm of weight decreased and margin increased. Also, elements of weight vector changed.
15 times norm of weight increased and margin decreased. Also, elements of weight vector changed.
0 times norm of weight and margin did not change.
```

All in all, when we remove one of the support vectors, norm of weight decreased and margin increased generally (36 out of 51 times). Also, elements of weight changed. Thus, hyperplane changed because weight is perpendicular to the hyperplane. When elements of weight changes, it means hyperplane changes. 15 out of 51 times, norm of weight increased and margin decreased. Also, elements of weight changed. It also means that hyperplane changed.

BONUS TASK:

In the bonus task, we are asked to use the toy data set from slides and implement a hard margin SVM using a QP-solver CVXOPT.

$$\min_{b,w} \frac{1}{2} w^T w$$

$$\text{subject to: } y_n(w^T x_n + b) \geq 1, \forall n$$

QP-solver, asks parameters in the format like below.

$$\min_{u \epsilon R^q} \frac{1}{2} u^T Q u + p^T u$$

subject to: $Au \geq c$

$$u^* = \begin{bmatrix} b^* \\ w^* \end{bmatrix} \leftarrow \text{QP(Q, p, A, c)}$$

$$Q = \begin{bmatrix} 0 & 0 \\ 0^d & I_d \end{bmatrix},\ p = 0_{d+1},\ A = -\begin{bmatrix} y1 & y1x1^T \\ .. & .. \\ yn & ynxn^T \end{bmatrix},\ c = -\begin{bmatrix} 1 \\ 1 \\ .. \\ 1 \end{bmatrix}$$

So, we arrange Q, p, A and c accordingly. Just as in the slides,

Note that, QP solver accepts the constraint in $-Au \leq -c$ format. So, we needed to multiply A and c with -1 as input.

```
     pcost       dcost       gap    pres   dres
 0:  3.2653e-01  1.9592e+00  6e+00  2e+00  4e+00
 1:  1.5796e+00  8.5663e-01  7e-01  2e-16  1e-15
 2:  1.0195e+00  9.9227e-01  3e-02  4e-16  9e-16
 3:  1.0002e+00  9.9992e-01  3e-04  2e-16  2e-15
 4:  1.0000e+00  1.0000e+00  3e-06  2e-16  1e-15
 5:  1.0000e+00  1.0000e+00  3e-08  1e-16  8e-16
Optimal solution found.
weight: [ 1.00000001 -1.00000001]
bias: -1.0000000134075104
Train classification accuracy: 100.0
```