**CMPE 462 – PROJECT 2**

**Implementing an SVM Classifier**

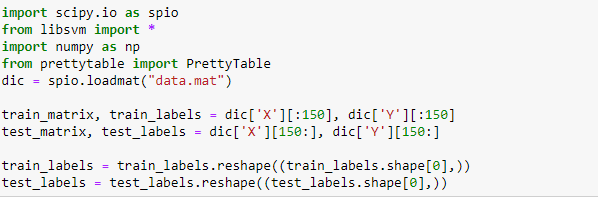
* **Student ID1:** 2014400051
* **Student ID2:** 2018400291
* **Student ID3:** 2019705072

**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.

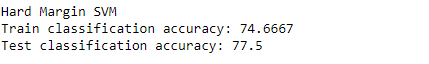


**INTRODUCTION:**

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.



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.

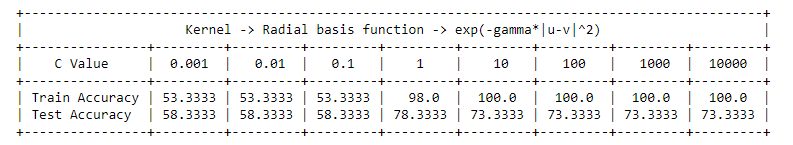
**TASK 2:**

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 C-values, and try different kernel functions.

Part 1: Fixed Kernel Functions, Different C-Values

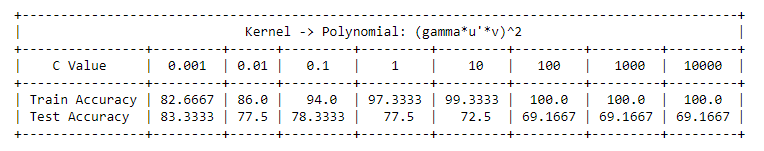
In part 1, we try different fixed kernel functions and different C-values for each fixed kernel functions.

1. Kernel function: Radial Basis Function



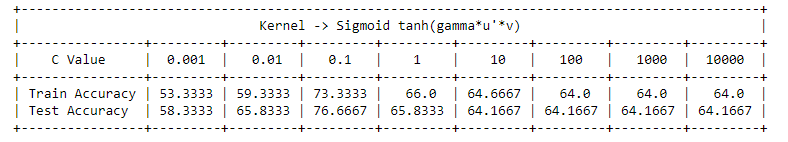
For the kernel, radial basis function, we observe that when C value increases, train and test accuracies increase. For C values between 1 and 10, model reaches the 100% training accuracy and 73.3% test accuracy, keeps constant for increased C values that we tried.

1. Kernel function: Quadratic Polynomial Function



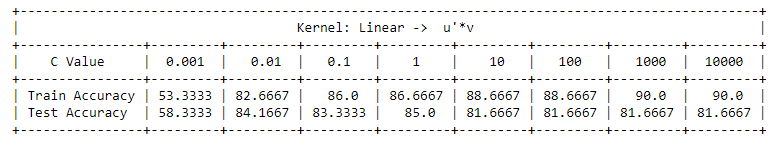
For the kernel, quadratic polynomial, function, we observe that when C value increases, train and test accuracies increase. For C values between 10 and 100, model reaches the 100% training accuracy and 69.17% test accuracy, keeps constant for increased C values that we tried.

1. Kernel function: Sigmoid Tanh Function



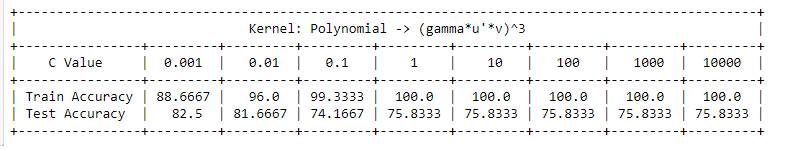
For the kernel, sigmoid tanh, function, we observe that when C value increases, train accuracy increases while C value reaches to a value between 0.1 and 1, with that value train accuracy has suddenly decreased. Then, it follows the decreasing trend until C value between 10 and 100. Also, same observation is valid for test accuracy, too. Finally, model reaches the 64% training accuracy and 64.17% test accuracy at C-value somewhere between 10 and 100, keeps constant for increased C values that we tried.

1. Kernel function: Linear Function



For the kernel, linear, function we observe that when C value increases, train and test accuracies increase. 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.

1. Kernel function: Cubic Polynomial Function

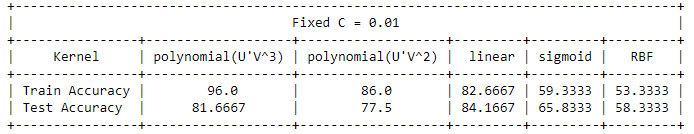


For the kernel, cubic polynomial, function we observe that when C value increases, train accuracy increases, but test accuracy decreases. For C values between 0.1 and 1, model reaches the 100% training accuracy and 75.83% test accuracy, keeps constant for increased C values that we tried.

Part 2: Fixed C-Values, Different Kernel Functions

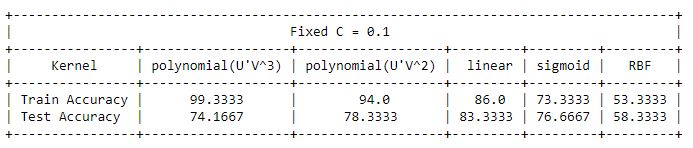
In part 2, we try different fixed C-Values and different kernel functions for each fixed C-Values.

1. Fixed C = 0.01



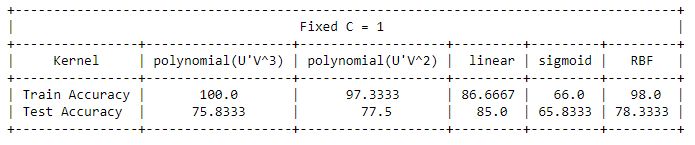
For the fixed C= 0.01, the highest train accuracy is given by the kernel cubic polynomial function, and highest test accuracy is given by the kernel linear function among the 5 different kernel functions shown in the above table.

1. Fixed C = 0.1



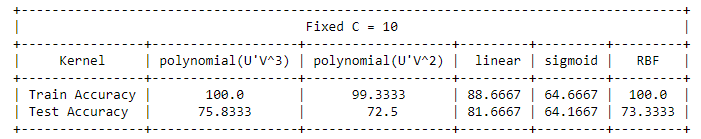
For the fixed C= 0.1, the highest train accuracy is given by the kernel cubic polynomial function, and highest test accuracy is given by the kernel linear function among the 5 different kernel functions shown in the above table.

1. Fixed C = 1



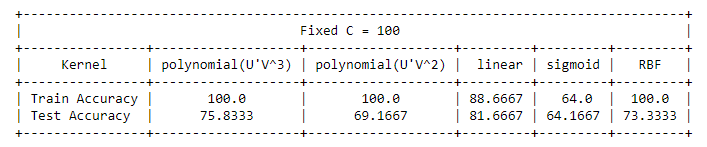
For the fixed C=1, the highest train accuracy is given by the kernel cubic polynomial function, and highest test accuracy is given by the kernel linear function among the 5 different kernel functions shown in the above table.

1. Fixed C = 10



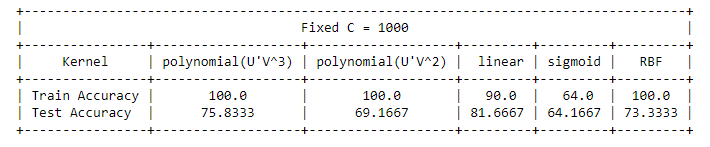
For the fixed C=10, the highest train accuracy is given by the kernels both cubic polynomial function and RBF, and highest test accuracy is given by the kernel linear function among the 5 different kernel functions shown in the above table.

1. Fixed C = 100



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

1. Fixed C = 1000

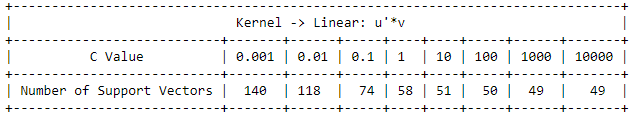


For the fixed C=1000, the highest train accuracy is given by the kernels cubic polynomial function, quadratic polynomial function and RBF, and highest test accuracy is given by the kernel linear function among the 5 different kernel functions shown in the above table.

**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.



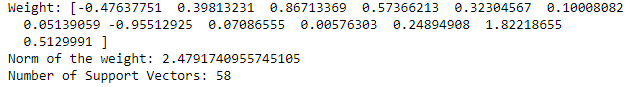
C value in SVM optimization, restricts how much of the data points are tolerated to violate of the constraints. If we increase C-value, a greater penalty is applied on violators and less data points are tolerated. So, SVM margin will be narrower. Thus, less data points are support vectors.

**TASK 4:**

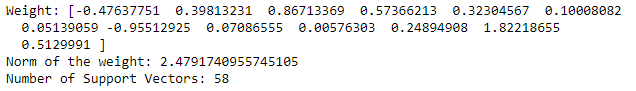
In this task we want to investigate the relationship between the hyperplane and support vectors and a data point that is not a support vector.

For this purpose, we used linear function as a kernel function. 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.

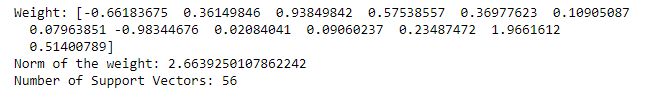


Removing one data point that is not a support vector



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



Weights, norm of the weight and number of support vectors have been changed after removing one of the support vectors. It causes a movement on the hyperplane, and transitively margin has changed, one should expect that margin increases, because one of the restrictions around hyperplane has been removed. The result is not surprising, norm of the weight increased, so margin is increased. Although we removed only one support vector, in the new model we have 2 missing support vectors compared with our initial model. This is also possible, hyperplane is completely different now.

**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.

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

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

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