Convex Optimization for Linear Support Vector Machines

Mihir Kapadia*
Technische Universiteit Delft

***Abstract*—Linear Support Vector Machines (SVM) are classification problems in the field of pattern recognition and machine learning which deal with segregation of objects into classes based on a given set of attributes. This introduces the notion of separation or discrimination of the data, which is generally equivalent to a set of inequalities. The SVM problem is then to identify the best hyperplane that classifies or discriminates the data points. Owing to the open-ended nature of the inequalities, the optimization problem becomes non-analytical leading to the need of complex recursive solvers. In this paper, the standard SVM is posed as a convex optimization problem. This problem is then solved using the first principles of optimization using Primal and Dual formulations. In addition various formulations of the standard problem are provided. The paper discusses the solutions obtained from off-the-shelf solvers as well as algorithmic minimization methods. Results from the Barrier method provide optimal yet convincing results, which are compared for the Primal and Dual Problem formulations.**

***Index Terms*—support vector machines, classification, convex optimization**

I. Introduction

In the paradigm of classification problems, support vector machines (SVM) are heuristics for multi-class separation. These are one of the most powerful supervised learning methods which are non-probabilistic classifiers and which rely on classical distance-based metrics. In Linear SVMs, a single data point is considered as a p-dimensional vector. A set of such points are separated using a p-1 dimensional hyperplane. The choice of a hyperplane can be guided by the largest possible separation achievable with most correct classification.

The goal of an SVM algorithm is to provide a set of hyperplanes which separate efficiently the data points while maximizing the gap between the boundaries of each class. This problem can be described as a set of linear inequalities in the order of dimension as that of the data. In this paper, a 2-class classification problem is tackled using SVMs, involving data set in a 2-dimensional plane. The margin of the classifier represents the minimum distance between the points of both the classes to the common separating hyperplane. The dataset is explained in Section II. Then, a formal convex problem definition for the given classification task with some alternative representations is provided in Section III. Thereafter in Section IV, various optimization approaches are presented including off-the-shelf solver, as well as an interior point method. In Section V, the results from the algorithms are discussed and compared for performance. Finally, in Section VI, we conclude the report with the findings.

* correspondence email - M.Kapadia-1@student.tudelft.nl, student no. : 5096278

Note: Submitted in the partial fulfillment of the course EE4530 - Applied Convex Optimization

II. Data Set

The data set provided for the experiment consists of 2 sets of points. One of the sets consisting of 100 points is serves as the training set and the other consisting of 900 points is used as a validation set. The points are in R^2 with each point having a label defining its class. The class labels are -1 and +1. The point data set is represented as $(x_1, y_1), \ldots, (x_n, y_n)$ Where y_i is either -1 or +1 indicating the class and x_i can be seen as a 2-dimensional coordinate location in R^2.

III. Convex Problem Formulation

As discussed in section I, the goal of the SVM problem is to find a linear hyperplane separating the two classes. Consider a point x in R^2. A hyperplane satisfying x can be described by an affine function as -

$$H : \{x | w^T x + b = 0\} \tag{1}$$

Where w represents the normal vector to the hyperplane and b represents residuals or offsets. Consider a data set with linearly separable classes. In such a case, there exists supporting hyperplanes to (1) limited by a certain $\gamma > 0$ which can be combined as

$$y_i(w^T x_i + b) \geq \gamma \quad i = 1, \ldots, n \tag{2}$$

These represent slabs adjacent to the hyperplane (1) with a unit residual. This slab ensures correct classification of the points. The goal of SVM is to maximize this margin. This margin is defined as a Euclidean distance measure between the closest points to the hyperplane. The margin is given by $2\gamma/||w||$, where $||w||$ is the norm of the normal vector w. $2/||w||$ represents the distance between the hyperplanes. This is the geometric margin, since it incorporates the condition of normal vector specific to the hyperplane, such that in maximizing γ the normal vector cannot go to infinity. At marginal points, the equality condition in (2) above holds.

A. Primal Objective Formulation

The ideal scenario is then to maximize the margin subject to the inequalities (2) -

$$\begin{aligned} \max_{w,b} \quad & \frac{2\gamma}{||w||_2} \\ \text{s.t.} \quad & y_i(w^T(x_i+b)) \geq \gamma \qquad i = 1, \ldots, n \end{aligned} \tag{3}$$

Taking an arbitrary $\gamma = 1$, and rewriting the above as a minimization problem we have –

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||_2^2 \\ \text{s.t.} \quad & y_i(w^T(x_i+b)) \geq 1 \qquad i = 1, \ldots, n \end{aligned} \tag{4}$$

This is the standard Convex formulation for the Linear Support Vector Machine Problem. As can be seen, the problem is a QP in the variable w and b.

1) Soft-Margin SVM [with slack variables]: The above formulation can be modified to suit better representation in multiple ways. One of the ways is to introduction of a slack variable. This is given as -

$$\begin{aligned} \min_{w,b,u} \quad & \frac{1}{n}\ 1^T z \\ \text{s.t.} \quad & y_i(w^T(x_i+b)) \geq 1 - z \\ & z \leq 0 \qquad\qquad i = 1, \ldots, n \end{aligned} \tag{5}$$

with variable w, b, z. where z represents the slack variable. This heuristic can be interpreted as minimizing misclassification of points. The variable z presents for the margin of mis-classification. Alternatively, combining the objectives of the above formulations gives a new problem formulation as follows -

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||_2^2 + \frac{C}{\gamma n}\ 1^T z \\ \text{s.t.} \quad & y_i(w^T(x_i+b)) \geq 1 - z \\ & z \leq 0 \qquad\qquad i = 1, \ldots, n \end{aligned} \tag{6}$$

Here, the parameters of C and γ represent the regularization parameters used to penalize and moderate the weight of the slacks appropriately. Too large a value of C yields a strict margin while small values of C ignores the importance of separating the data and allows for large slack. The parameter γ has the opposite effect and is introduced for a stable formulation for descent methods.

B. Dual Problem Formulations

The Primal problem is a QP with inequality constraints given by (6). The Dual of the problem can be derived by Lagrange Formulation. The Lagrangian Dual function is -

$$\begin{aligned} L(w,b,z,\lambda,\alpha) = & \frac{1}{2}\|w\|^2 + \frac{C}{\gamma n}\ 1^T z + \sum_{i=1}^{n} \alpha_i(-z_i) \\ & + \sum_{i=1}^{n} \lambda_i(1 - y_i(w^T(x_i+b)) - z_i) \end{aligned} \tag{7}$$

Primal and Dual solutions are related as -

$$p^* = \inf_{w,b,z} \sup_{\alpha,\lambda>0} L \ \geq \ \sup_{\alpha,\lambda>0} \inf_{w,b,z} L = d^* \tag{8}$$

Constraints are satisfied for $w = b = 0$ and $z - \mathbf{1}$. Therefore there exists strong duality by slater's conditions. We have $p^* = d^*$. The dual function can be obtained from KKT conditions – minimizing the dual objective over w and b and plugging the resultant w^* and b^* values in the objective.

$$\begin{aligned} \frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \lambda_i y_i x_i = 0 \qquad & \frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \lambda_i y_i = 0 \\ w^\star = \sum_{i=1}^{n} \lambda_i y_i x_i \qquad & \sum_{i=1}^{n} \lambda_i y_i = 0 \end{aligned} \tag{9}$$

Upon substitution and simplification, the Dual Function g in λ becomes

$$g(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i^T x_j \tag{10}$$

Then the Dual Problem is

$$\begin{aligned} \max_{\lambda} \quad & \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i^T x_j \\ \text{s.t.} \quad & \lambda_i \geq 0, \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n} \lambda_i y_i = 0, \quad i = 1, \ldots, n \end{aligned} \tag{11}$$

The above problem can be simplified further to suit a QP representation as -

$$\begin{aligned} \max_{\lambda} \quad & 1^T\lambda - \frac{1}{2}||\sum_{i,j} \lambda_i y_i x_i||_2^2 \\ \text{s.t.} \quad & \lambda_i \geq 0, \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n} \lambda_i y_i = 0, \quad i = 1, \ldots, n \end{aligned} \tag{12}$$

With strong duality, we have complementary slackness as -

$$\lambda_i^\star(1 - y_i(w^{\star T} x_i + b^\star - z_i)) = 0, \quad i = 1, \ldots, n \tag{13}$$

Also, from differentiation of Dual function, we have $\alpha_i^* = (C/n) - _i^*$. Therefore, we get the conditions on dual variables as -

$$\alpha_i^* z_i^* = (\frac{C}{n} - \lambda_i^*) z_i^* = 0 \tag{14}$$

From Complementary slackness, we can combine the inequality constraints to a single inequality on the Langrangian variable λ. give the form -

$$\begin{aligned} \min_{\lambda} \quad & -1^T\lambda + \frac{1}{2}||\sum_{i,j} \lambda_i y_i x_i||_2^2 \\ \text{s.t.} \quad & 0 \leq \lambda_i \leq \frac{C}{\gamma n} \quad i = 1, \ldots, n \end{aligned} \tag{15}$$

IV. Optimization Algorithms

In this section, a series of algorithms are proposed in order to solve the problems posed in the previous section. Various problem formulations are tested including the Primal and Dual Objective functions.

A. CVX Solver

Starting with solution from off-the-shelf solver, the SVM problem is defined with solver. The code is presented in Appendix A. The software-package used is CVX with MATLAB. The solver implements infeasible path-finding algorithms much to the tune of the method discussed later for finding a solution in alternate way.The class of algorithms used is SDPT3. It also relies on Newton step method, specifically the HKM step direction and converts the problem to standard form to solve for the optimal value. The results are seen in Fig. 1. The Table I shows the optimization results with the runtime parameters. These shall act as benchmark for the comparisons. The code for the implementation is attached in Appendix A.

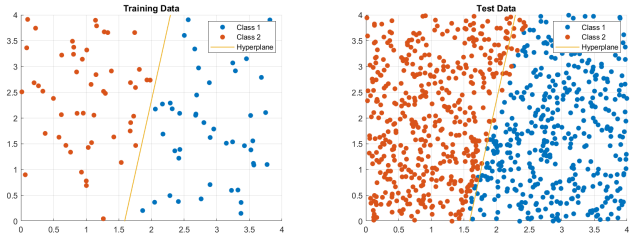

Fig. 1. Hyperplane Classification using CVX

TABLE I
CVX Results

Parameter	Value
w_1	11.2556
$w2$	-1.9739
b	-17.8885
Time	0.43s
Iterations	22
Gap	1.49e-08

B. Algorithm for Primal & Dual Problems

The Primal Problem given by (6) is a Quadratic Problem with inequality constraints. Similarly, the Dual Problem given by (12) can be transformed to a Quadratic form with sufficient modifications. In lieu of this observation, the solutions to both Primal and Quadratic Problem can be found by applying a Gradient method on inequality constrained QP. However, first the problems are converted to observe the QP form in the next section.

1) Problem Reformulation: Consider the standard form of QP in s with inequality constraints -

$$\begin{aligned} \min_{\lambda} \quad & \frac{1}{2} s^T Q s + p^T s \\ \text{s.t.} \quad & As \leq b \end{aligned} \tag{16}$$

The goal here is to obtain equivalent formulations of the matrices (Q, p, A, b) for both Primal and Dual problems.

For Primal Problem of (6) absorbing w, b to a single variable $w = [b \quad w]^T$. This requires an additional reformulation on the data X - addition of a column of ones - $X = [1_n X]$.

$$Q = \begin{bmatrix} I_{d+1} & 0_{d+1xn} \\ 0_{nxd+1} & 0_{nxn} \end{bmatrix} \qquad A = \begin{bmatrix} -diag(y)X & -I_n \\ 0_{nxd+1} & -I_{nxn} \end{bmatrix}$$

$$p = \begin{bmatrix} 0_{d+1} \\ \frac{C}{\gamma n} 1_n \end{bmatrix} \qquad b = \begin{bmatrix} -1_n \\ 0_n \end{bmatrix}$$

Similarly for Dual Problem of (15), we get

$$Q = diag(y) X X^T diag(y) \qquad A = \begin{bmatrix} -I_n \\ -I_{nxn} \end{bmatrix}$$

$$p = 1_n \qquad b = \begin{bmatrix} -\frac{C}{\gamma n} 1_n \\ 0_n \end{bmatrix}$$

With the above formulations, the generalised QP of (16) can be solved using a single algorithm. The form of QP for both the Primal and Dual problems is easily solvable using Newton's Descent method. Since the problems are continuous, they are differentiable. Next we explore the interpretations from the derived constraints with the help of KKT conditions.

C. Interpretation via KKT Conditions

With the conversion of the Primal and Dual Problems to the new form, the constraint minimizations are restricted to box constraints, which lends them cheaper to optimize. In addition, (13) and (14), suggest that the optimal primal solution equals the dual solution. By strong duality -

$$p^* = d^* \tag{17}$$

The primal optimal value can be obtained from the solution to the dual problem, λ^* as -

$$w^* = \sum \lambda_i^* y_i x_i \tag{18}$$

The consequences of complementary slackness for the original problem can be summarized as follows.

- if $y_i w^{*T} x_i + b^* > 1$, then $z_i^* = 0, \lambda_i^* = 0$
- if $y_i w^{*T} x_i + b^* < 1$, then $z_i^* > 0, \lambda_i^* = C/n$
- if $\lambda_i^* = 0$ then $z_i^* = 0, \Rightarrow (y_i(w^{*T} x_i + b^*)) > 1$
- if $\lambda_i^* \in (0, C/n) then z_i^* = 0, \Rightarrow (1 - y_i(w^{*T} x_i + b^*) = 0$

In order to obtain b^*, we use complementary slackness. From the final condition above

$$\begin{aligned} y_i(w^{*T} x_i + b^*)) &= 1 \\ b^* &= y_i - x_i^T w^* \end{aligned} \tag{19}$$

D. Barrier Method

An interesting observation of the formulation suggests the application of the log-barrier method by manipulating the problems in (6) and (15), which can now be defined with (16). Applying log barrier to the QP we have -

$$\phi = \quad t(\frac{1}{2} s^T Q s + p^T s) + I_-(As - b) \tag{20}$$

where $I_-(s)$ represents $-\sum log(s)$. The Gradient and Hessian of ϕ is :

$$\nabla \phi = t(Q^T s + p) - \sum \frac{w}{b - w^T s}$$

$$\nabla^2\phi = tQ + \sum \frac{ww^T}{(b - w^T s)^2}$$

The objective of Primal and Dual problems is minimization of ϕ which can be achieved using Newton's Method. Newton Decrement is given by -

$$\lambda = \sqrt{(\nabla\phi)^T(\nabla^2\phi)^-1(\nabla\phi)}$$

The pseudocode is derived from Chapter 12. of 1.The outer loop represents the Barrier Method Loop for line update step. The inner loop represents the Newton Descent Loop with line search according to Newton's Decrement Direction. The initial step is called the centering step as it effectively computes the central path for the QP. The algorithm for the Barrier Method and function of Newton Method are available in Appendix D and E. The code for the Primal and Dual Problems are called using the codes attaches in Appendix B and C, respectively.

Algorithm 1 Barrier Method

Given starting point for x, t, μ, ϵ
while $m/t > \epsilon$ **do**
1. Compute x^* by Newton Method - Minimizing ϕ.
2. Update x. $x := x^*$
3. Increase t. $t := \mu t$

end

The Barrier Method Parameters for the Primal and Dual Problems are summarized in II and III respectively.

TABLE II
PRIMAL PROBLEM SETTINGS

Parameter	C	γ	μ	ϵ
Value	40	0.001	10	$1e^{-7}$

TABLE III
DUAL PROBLEM SETTINGS

Parameter	C	γ	μ	ϵ
Value	150	0.05	10	$1e^{-7}$

E. Convergence Analysis

Owing to the quadratic approximation by Newton's descent, the Barrier method achieves much faster and tractable convergence.

- No. of Iterations : The algorithm terminates with $\phi^* - p^* \leq \epsilon$, where ϵ is the tolerance criteria. This follows from the stopping criteria $\phi^* - p^* \leq m/t$, where m is the size of the dataset.
- Choice of μ : Choice of μ is subject of design preference. Small μ means a feasible point for Newton's method and hence lower iterations per inner loop. However, this also means that the gap reduction towards achieving the optimal is very small, hence increasing the number of outer iterations. Large μ means faster iterative update, meaning the subsequent iterates are farther away implying large jumps. This forces the inner loop for larger runs in order to bring the solution to feasible set.

V. RESULTS

This section discusses the results from the Barrier Method as applied on the Primal and Dual Problems.

1) Classifier Results for the Primal Dual Problem.: The classifier results for the training and test datasets for the Primal problem are shown in Fig. 2 and for the Dual problem are shown in Fig. 3. The parameter results can be found in IV. As visible, the Primal and Dual Problems successfully solve the SVM Classification problem with good separation of data points almost equivalently. The hyperplane separation is skewed in the case of dual solution owing to parameter variations. This can be adjusted for complete equivalence, since as stated, strong duality holds.

TABLE IV
CVX RESULTS

Parameter	**Primal**	**Dual**
w_1	11.2556	463.5954
$w2$	-1.9739	-140.3652
b	-17.8885	-605.4312
Time	8.1552s	35.695
Iterations	11	11
Infeasibility Gap	22.5	20

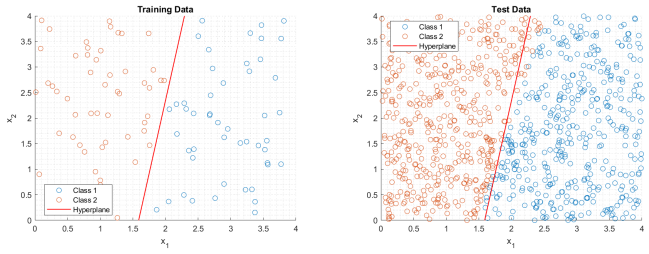

Fig. 2. Hyperplane Classification using Barrier Method on Primal Problem

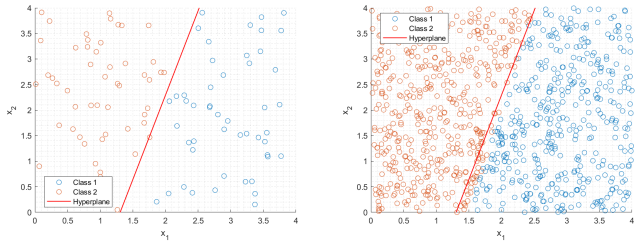

Fig. 3. Hyperplane Classification using Barrier Method on Dual Problem

2) Comparison between Primal and Dual Algorithms: . To analyse the performance of the Barrier Method on Primal and Dual Problems, we may compare the results obtained from the two solution by numerous meta-parameters such as the iterative objective value, slack variable and the tolerance gap amongst successive steps. This analysis is tabulated here. Fig. 4 is a comparative look at objective value reductions. The plots are semilog in y. As seen, both take 11 iterations to convergence suggesting equivalence in objectives. However,

the dual problem sees a rather slower reduction in objective value than the primal one. This is can be attributed to the parameter settings. In addition, the difference also suggests infeasibilities in finding optimal Newton decrement direction for the dual problem.

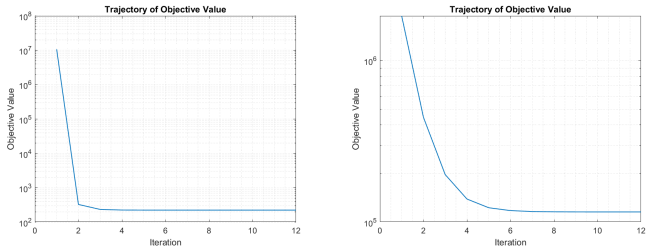

Fig. 4. Comparison of Objective Value Trajectories

Secondly, the relationship between slack reductions for the primal and dual problems can be seen in Fig. 5. This semilog graph in y shows a different picture, while the primary problem slack reduces gradually, the dual slack quickly rather quickly. This is because of available freedom of the slack variable from the objective. Finally, it may be worth noting that the classifiers

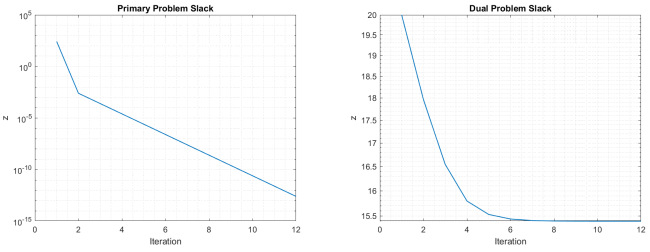

Fig. 5. Comparison of Slack Variables Trajectories

presented here need to start with feasible points. Further, the proposed Barrier Method requires strict feasibility rendering it difficult to use in practical cases where feasible points are difficult to find. This can be subverted, by an alternative to the Newton Method used for optimization. This is by inclusion of an infeasible start condition.

VI. Conclusion

From the previous section, it can be seen that the Barrier Method provides reliable results for the SVM classification problem. The Primal and Dual Problem solutions however, differ in scale due to different formulations. Compared to the benchmark of the CVX solver, we see that the Barrier Method implementation is suboptimal in limiting sense. It achieves the classification required with the Primal solution equivalent to the CVX benchmark, however, the dual problem falters in results by a margin. Comparing the time of computation and resources, it can be easily seen that the CVX solver outperforms the implementation by a very large margin. In addition, we see the dependence of the result on the parameters defined in the problem.

References

[1] S. Boyd and L. Vandenberghe, Convex optimization. Cambridge university press, 2004.

APPENDIX

A. CVX Solver

```
clear all
clc
close all

%% SVM Problem
load('linear_svm.mat')
n = length(X_train);

cvx_begin
variables w1(2,1) b1(1)
   minimize (0.5*w1'*w1)
   subject to
   ones(1,100) - labels_train'.*(w1'*X_train' + b1*ones(1,100)) <= 0;
cvx_end

%% Plotting Function


% Plot of Classifier Training Data
class1 = (labels_train > 0);
class2 = (labels_train < 0);
figure
scatter(X_train(class1,1),X_train(class1,2),'filled','DisplayName','Class 1')
hold on
scatter(X_train(class2,1),X_train(class2,2),'filled' ,'DisplayName','Class 2')
x1 = linspace(0,4,100);
xlim([0,4]); ylim([0,4]);

w = w1; b = b1;
x2 = -(b + w(1,1) * x1 ) /w(2,1);
plot(x1 ,x2,'DisplayName','Hyperplane','Linewidth',1);
title('Training Data')
grid on
legend
saveas(gcf,'cvx_train.png')

% Plot of Classifier Test Data
class1 = (labels_test > 0);
class2 = (labels_test < 0);
figure
scatter(X_test(class1,1),X_test(class1,2),'filled','DisplayName','Class 1')
hold on
scatter(X_test(class2,1),X_test(class2,2),'filled','DisplayName','Class 2')

xlim([0,4]); ylim([0,4]);

w = w1; b = b1;
x2 = -(b + w(1,1) * x1 ) /w(2,1);
plot(x1 ,x2,'DisplayName','Hyperplane','Linewidth',1);
title('Test Data')
grid on
legend
saveas(gcf,'cvx_test.png')
```

B. Primal Problem

```
% test
close all
clear all
clc

% Loading the data
```

```
load linear_svm.mat

% The training data
X_train = X_train;
y_train = labels_train;
% The testing data
X_test = X_test;
y_test = labels_test;


%% Primal Problem Solution
tic
C = 40;
gamma = 0.001;
[n,d]=size(X_train);

% Formulating Standard QP form
X = [ones(n,1) X_train];
Q = [eye(d+1) zeros(d+1,n); zeros(n,d+1) zeros(n,n)];
p = [zeros(d+1,1);(C/(gamma*n))*ones(n,1)];
b = [-1*ones(n,1); zeros(n,1)];
A = [-diag(y_train)*X -eye(n);zeros(n,d+1) -eye(n)];

% Barrier Method Initializations
w0_primal= [zeros(d+1,1); (C/(1.5*gamma*n)).* ones(n,1)];
mu = 10;
tol = 0.0000001;
[w_primal,w_hist,tol_gap,obj] = newton_barrier(Q,p,A,b,w0_primal,mu,tol);

% Hyperplane Results of the Dual Solution
w_p = w_primal(1:3);

w_h = w_hist(1:3,:);

toc

%% Plot Functions

x1 = linspace(0,4,100);
x2 = -(w_p(1,1) + w_p(2,1) * x1 ) /w_p(3,1) ;

% Response on Training Data
figure
class1 = (labels_train > 0);
class2 = (labels_train < 0);
scatter(X_train(class1,1),X_train(class1,2),'DisplayName', 'Class 1')
hold on
scatter(X_train(class2,1),X_train(class2,2),'DisplayName', 'Class 2')
xlabel('x_1'); ylabel('x_2');
xlim([0,4]); ylim([0,4]);
plot(x1 ,x2 , 'r' ,'linewidth',1,'DisplayName', 'Hyperplane');
grid minor
title('Training Data')
legend('Location','best')
saveas(gcf,'p_train.png')

% Response on Training Data
figure
class1 = (labels_test > 0);
class2 = (labels_test < 0);
scatter(X_test(class1,1),X_test(class1,2),'DisplayName', 'Class 1')
hold on
scatter(X_test(class2,1),X_test(class2,2),'DisplayName', 'Class 2')
xlabel('x_1'); ylabel('x_2');
xlim([0,4]); ylim([0,4]);
plot(x1 ,x2 , 'r' ,'linewidth',1,'DisplayName', 'Hyperplane');
grid minor
```

```
title('Test Data')
legend('Location','best')
saveas(gcf,'p_test.png')

% Plot of Primal Problem Slack
figure
semilogy(mean(w_hist(4:end,:)),'linewidth',1);
grid minor
xlabel('Iteration'); ylabel('z');
title('Primary Problem Slack')
saveas(gcf,'p_w.png')

% Plot of Objective Value
figure
semilogy(obj ,'linewidth',1);
grid minor
xlabel('Iteration'); ylabel('Objective Value');
title('Trajectory of Objective Value')
saveas(gcf,'p_obj.png')

% Plot of Gaps at Each step
figure
semilogy(tol_gap ,'linewidth',1);
grid minor
xlabel('Iteration'); ylabel('Tolerace Gap m/t');
title('Trajectory of Tolerance Gap')
saveas(gcf,'p_gap.png')
```

C. Dual problem

```
% test
close all
clear all
clc

% Loading the data
load linear_svm.mat

% The training data
X_train = X_train;
y_train = labels_train;
% The testing data
X_test = X_test;
y_test = labels_test;


%% Dual Problem SOlution
tic
C = 150;
gamma = 0.05;
[n,d]=size(X_train);

% Formulating Standard QP form
Q = diag(y_train)*X_train*(X_train')*diag(y_train);
p = -1*ones(n,1);
b = [(C/(gamma*n)).*ones(n,1); zeros(n,1)];
A = [eye(n);-1*eye(n) ];

% Barrier Method Initializations
s0_dual= (C/(1.5*gamma*n)).* ones(n,1);
mu = 10;
tol = 0.0000001;
[s_dual,s_hist,tol_gap,obj] = newton_barrier(Q,p,A,b,s0_dual,mu,tol);

% Hyperplane Results of the Dual Solution
```

```
w_d = X_train'*diag(y_train)*s_dual;

b    = mean(y_train - X_train*w_d);

toc

%% Plot Functions

x1 = linspace(0,4,100);
x2 = -(b + w_d(1,1) * x1 ) /w_d(2,1) ;

% Response on Training Data
figure
class1 = (labels_train > 0);
class2 = (labels_train < 0);
scatter(X_train(class1,1),X_train(class1,2),'DisplayName', 'Class 1')
hold on
scatter(X_train(class2,1),X_train(class2,2),'DisplayName', 'Class 2')
xlabel('x_1'); ylabel('x_2');
xlim([0,4]); ylim([0,4]);
plot(x1 ,x2 , 'r' ,'linewidth',1,'DisplayName', 'Hyperplane');
grid minor
legend('Location','best')
saveas(gcf,'d_train.png')

% Response on Training Data
figure
class1 = (labels_test > 0);
class2 = (labels_test < 0);
scatter(X_test(class1,1),X_test(class1,2),'DisplayName', 'Class 1')
hold on
scatter(X_test(class2,1),X_test(class2,2),'DisplayName', 'Class 2')
xlabel('x_1'); ylabel('x_2');
xlim([0,4]); ylim([0,4]);
plot(x1 ,x2 , 'r' ,'linewidth',1,'DisplayName', 'Hyperplane');
grid minor
legend('Location','best')
saveas(gcf,'d_test.png')

% Plot of Primal Problem Slack
figure
semilogy(mean(s_hist),'linewidth',1);
grid minor
xlabel('Iteration'); ylabel('z');
title('Dual Problem Slack')
saveas(gcf,'d_w.png')

% Plot of Objective Value
figure
semilogy(obj ,'linewidth',1);
grid minor
xlabel('Iteration'); ylabel('Objective Value');
title('Trajectory of Objective Value')
saveas(gcf,'d_obj.png')

% Plot of Gaps at Each step
figure
semilogy(tol_gap ,'linewidth',1);
grid minor
xlabel('Iteration'); ylabel('Tolerace Gap m/t');
title('Trajectory of Tolerance Gap')
saveas(gcf,'d_gap.png')
```

D. Barrier Method

```
function [s_sol,s_hist,tol_gap,obj] = newton_barrier(Q,p,A,b,s0,mu,tol)

% Barrier Method Initializations
t = 1;
total_iters = 1000;
m = size(A,1);
s = s0;
s_hist = s;
obj = 0.5*s0'*Q*s0+p'*s0;
tol_gap = [];

for i = 1 : total_iters
   s = newton_descent(s,t,Q,p,A,b,tol);

   obj = [obj 0.5*s'*Q*s + p'*s];
   s_hist = [s_hist s];
   tol_gap = [tol_gap m/t];

   % Explicit Check on achieved tolerance
   if (m/t) < tol
      break;
   else
      t = mu*t;
   end
end
s_sol = s;
fprintf('Barrier Iterations =%d \n',i);
```

E. Newton Method

```
function s_opt = newton_descent(s0,t,Q,p,A,b,tol)

% Initialization for current step
s_n = s0;
descent_iters = 1000;

for k = 1 : descent_iters
   % Computing Derivatives

   % Residual Vector
   vector = A*s_n - b;

   % Gradient Value
   g = t * (Q*s_n + p) - A' * (1./vector);

   % Hessian Value
   h = t*Q;
   for r = 1:size(A,1)
   h = h + (A(r,:)'*A(r,:))./(vector(r)^2);
   end

   % Computing Newton Step & Newton Gap
   newton_step = - h\g;
   % Descent Direction
   newton_direction = - g' * newton_step;
   factor = 1/(1+sqrt(newton_direction));

   % Descent Step Update along Newton Decrement Direction
   s_n1 = s_n + factor * newton_step;

   s_n = s_n1;
   % Checking Gap vs. Tolerance per descent step
   if newton_direction/2 < tol
    break;
   end
```

```
end
s_opt = s_n;
```