UNIVERSITY OF BALAMAND

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING

CPEN425

Neural Network Design

Submitted to: Dr. Issam Dagher

Submitted by: Tania Aoude (A2010001)

Hala Msalem (A2020119)

Submitted on: April 25, 2025

Table of Contents

Table of Figures

I. Introduction

This project consists of three problems: perceptron, classic SVM and dual SVM. The first method performs a weight updated based on whether the value of the output is equal to the target. This gives the proper set of weights that will correctly classify the data. The second method (direct method) is an optimization problem that solves for the best set of weights (including the bias) that will give the best separation line for the data. Finally, the last problem (indirect method) is also an optimization problem that solves for the best values of αi which are then used to calculate the appropriate set of weights.

II. Solution

1. Problem 1: Perceptron

In this method, a weight update is performed when there is an error and the value of the output is not equal to the target. The weights are updated based on the following formula:

$$wnew = wold + (learning\ rate)(target - output)x$$

Output O is equal to 0 when net is less than 0, otherwise it is equal to 1 where net is as follows:

$$net = w_1x_1 + w_2x_2 + - - - + w_nx_n + w_{n+1}$$

- Code

```
close all
clc
x=[-1 -1 1; -1 1 1; 1 -1 1; 1 1 1];
x1=x(:,1);
x2=x(:,2);
x3=x(:,3);
t=[-1;-1;-1;1];
trainingdata= table(x1,x2,x3,t);
[n,m]= size(x);
w=zeros(m,1);% intialize weights
alpha=1;% set learning rate
net=0;
O=0;

errorvector=zeros(1,n);% vector to store the error at each iteration
in
while true
```

```
for i=1:n

    fprintf('iteration:%d \n ',i)
    net=w(1)*x1(i)+w(2)*x2(i)+w(3)*x3(i); %calculate net

    if net<0 % value of output depending on if net is >0 or <=0
        O=-1;
    else
        O=1;
    end

    error=alpha*(t(i)-O);% calculate error
    errorvector(1,i)=error;
     if O~=t(i) %if output different from target update weights
        w(1)=w(1)+error*x1(i);
        w(2)=w(2)+error*x2(i);
        w(3)=w(3)+error*x3(i);

     end
end
if sum(errorvector)==0
    break %exit while when there is no error, otherwise continue for
another epoch
end
end
```

- Results

With the weights initialized to zero and alpha to 1,

trainingdata =

4×4 table

x1	x2	x3	t
-1	-1	1	-1
-1	1	1	-1
1	-1	1	-1
1	1	1	1

Figure 1: Training Data

For the first epoch, since an error occurs in the 1st iteration the weights are updated as shown in Figure 2,

```
iteration:1

net =

     0


O =

     1


error =

    -2
```

```
w =

     2
     0
     0


w =

     2
     2
     0


w =

     2
     2
    -2
```

Figure 2: Result of Weight Update of The First Iteration Epoch 1

Since no error occurs in the 2nd, 3rd and 4th iterations, the weights are not updated as shown below.

```
iteration:2

net =

    -2


O =

    -1


error =

     0
```

```
iteration:3

net =

    -2


O =

    -1


error =

     0
```

```
iteration:4

net =

     2


O =

     1
```

Figure 3: Result of Weight Update of the Last 3 Iterations Epoch 1

Because weight change occurred during the first epoch, the code repeats for another epoch until no weight change occurs. Since there was no error during the second epoch, the weights were not updated and the while loop was terminated.

```
iteration:1

error =

     0

iteration:2

error =

     0

iteration:3

error =

     0

iteration:4

error =

     0
```

Figure 4: Result of Weight Update of the 4 Iterations Epoch 2

The results match that of the solved example in class, and each iteration corresponds to a set of inputs (Input 1= (x1,x2) in the first row of the table and so on).

2. Problem 2: SVM

In this problem, we implement the primal form of a linear Support Vector Machine (SVM) to find the optimal decision boundary that separates two classes. The SVM attempts to maximize the margin between the two classes by solving a convex quadratic programming problem. This method directly solves for the weight vector w and bias b.

The optimization problem solved here is:

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

Subject to:

$$y_i(w^T x_i + b) \geq 1$$

or

$$-y_i(w^T x_i + b) \leq -1$$

Where:

- xi are the input vectors,
- yi ∈ $\{-1,1\}$ are the target labels

– b is the bias term
– w1 and w2 are the weights
– weights (wi) and b define the decision boundary

The goal is to find the hyperplane $wTx + b = 0$ with the maximum margin between the two classes.

To solve this using quadprog, we rewrite the optimization as:

$$min_z \frac{1}{2} z^T H z + f^T z$$

Subject to:

$$Az \le c$$

Where $z = [w1\ w2\ b]^T$

- Code

```
X = [3 7; 4 6; 5 6; 7 7; 5 8; 4 5; 5 5; 6 3; 7 4; 9 4];
y = [1 1 1 1 1 -1 -1 -1 -1 -1]'; % Transpose to make it a column
vector

% Number of data points
n = length(y);

H = [
    1 0 0;
    0 1 0;
    0 0 0
    ];

f = [0; 0; 0];

A = [
```

```
    -y(1)*X(1,1) -y(1)*X(1,2) -y(1);
    -y(2)*X(2,1) -y(2)*X(2,2) -y(2);
    -y(3)*X(3,1) -y(3)*X(3,2) -y(3);
    -y(4)*X(4,1) -y(4)*X(4,2) -y(4);
    -y(5)*X(5,1) -y(5)*X(5,2) -y(5);
    -y(6)*X(6,1) -y(6)*X(6,2) -y(6);
    -y(7)*X(7,1) -y(7)*X(7,2) -y(7);
    -y(8)*X(8,1) -y(8)*X(8,2) -y(8);
    -y(9)*X(9,1) -y(9)*X(9,2) -y(9);
    -y(10)*X(10,1) -y(10)*X(10,2) -y(10)
    ];



c = [-1; -1; -1 ; -1; -1; -1; -1; -1; -1; -1];

% Solve the quadratic programming problem
z = quadprog(H, f, A, c);

% Extract w1, w2, and b
w1 = z(1);
w2 = z(2);
b = z(3);

% Plot the data and the three lines
figure;
gscatter(X(:, 1), X(:, 2), y, 'rb', '+*'); % Plot data points
hold on;

% Set axis limits
xlim([0 10]); % Set x-axis limits from 0 to 10
ylim([0 10]); % Set y-axis limits from 0 to 10

% Plot the decision boundary (w1*x1 + w2*x2 + b = 0)
x_plot = linspace(0, 10, 100); % Adjusted x_plot range
y_plot_decision = (-w1 * x_plot - b) / w2;
```

```
plot(x_plot, y_plot_decision, 'k-', 'LineWidth', 2); % Black solid
line

% Plot the margin line (w1*x1 + w2*x2 + b = 1)
y_plot_margin_plus = (-w1 * x_plot - b + 1) / w2;
plot(x_plot, y_plot_margin_plus, 'g--', 'LineWidth', 1.5); % Green
dashed line

% Plot the margin line (w1*x1 + w2*x2 + b = -1)
y_plot_margin_minus = (-w1 * x_plot - b - 1) / w2;
plot(x_plot, y_plot_margin_minus, 'm--', 'LineWidth', 1.5); % Magenta
dashed line

title('SVM Decision Boundary and Margins');
xlabel('X1');
ylabel('X2');
legend('Class 1', 'Class -1', 'Decision Boundary', 'Margin (+1)',
'Margin (-1)');
hold off;
```

- Results

The resulting plot shows the data, decision boundary, and margins.

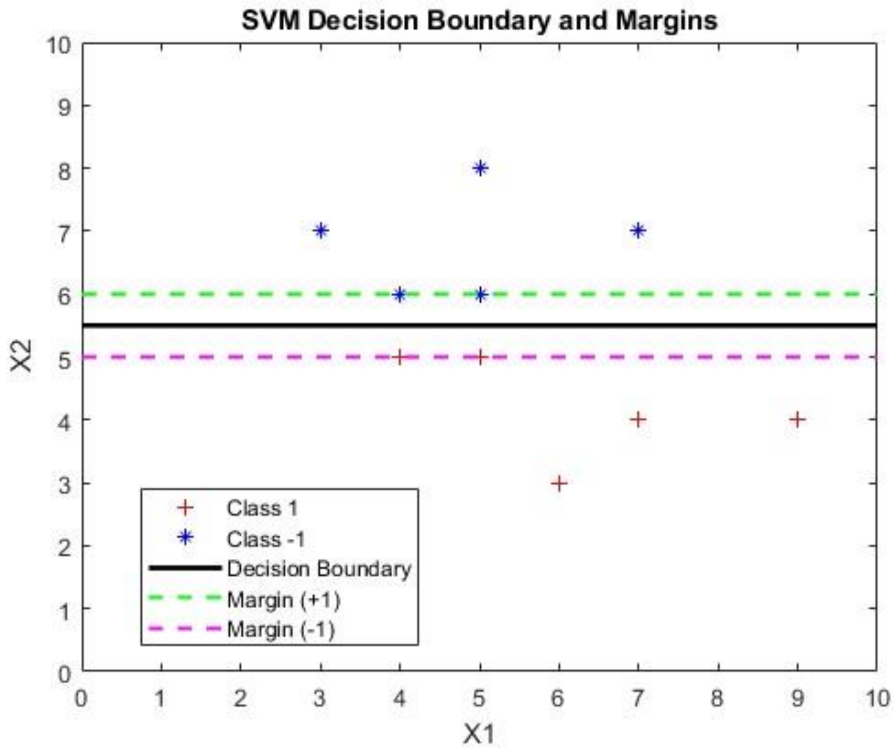

Figure 5: Results of SVM

We see in the graph above:

- The optimal separating hyperplane: $w_1x_1 + w_2x_2 + b = 0$
- The maximum margin lines: $w_1x_1 + w_2x_2 + b = \pm 1$

3. Problem 3: Dual SVM

In this problem, we use the **dual formulation** of the SVM, which allows us to find the solution by optimizing Lagrange multipliers αi instead of directly solving for w and b.

The dual problem is:

$$max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\ \alpha_j\ y_i\ y_j \langle x_i, x_j \rangle$$

Subject to:

$$\alpha_i \geq 0 \; and \; \sum_{i=1}^{n} \alpha_i \, y_i = 0$$

Where:

- αi are the Lagrange multipliers
- ⟨xi,xj⟩ is the dot product between input vectors
- yi∈ are the class labels
- Xi are the input vectors
- b is the bias term
- w1 and w2 are the weights

This is equivalent to:

$$min_z \, \frac{1}{2} z^T H z + f^T z$$

Subject to:

$$Az \leq c$$

$$Aeqz = ceq$$

$$zl \leq z \leq zu$$

Where**:**

$\boldsymbol{z} = [\alpha_1, \alpha_2, \dots, \alpha_n]^{\boldsymbol{T}}$

$w = \sum_i \alpha_i y_i x_i$

$b = 1 - \sum_{j=1}^{n} \alpha_j y_j (x_j . x_i)$

- Code

```
% Data
X = [3 7; 4 6; 5 6; 7 7; 5 8; 4 5; 5 5; 6 3; 7 4; 9 4];
y = [1 1 1 1 1 -1 -1 -1 -1 -1]'; % Transpose to make it a column
vector

% Number of data points
n = length(y);

% H matrix
H = (y * y') .* (X * X');

% f vector
f = -ones(1, n); % Note the row vector

% A matrix
A = zeros(1, n); % Or A = []

% c vector
c = 0; % Or c = []

% Aeq matrix
Aeq = y';

% ceq vector
ceq = 0;

% lb vector
zl = zeros(n, 1);

% ub vector
zu = 1000 * ones(n, 1);
```

```
% Solve the quadratic programming problem with all parameters
alpha = quadprog(H, f, A, c, Aeq, ceq, zl, zu);

% Extract support vectors (alpha > 1e-5)
support_vector_indices = find(alpha > 1e-5);
support_vectors = X(support_vector_indices, :);
support_vector_labels = y(support_vector_indices);
support_vector_alphas = alpha(support_vector_indices);

% Calculate w
w = zeros(1, size(X, 2)); % Initialize w
for i = 1:length(support_vector_indices)
    w = w + support_vector_alphas(i) * support_vector_labels(i) *
X(support_vector_indices(i), :);
end

% Calculate b using the formula
% Select a support vector that lies on the margin (w'x + b = 1)
i = 1; % Choose the first support vector
b = 1 - sum(alpha(support_vector_indices) .* support_vector_labels .*
(X(support_vector_indices, :) * X(support_vector_indices(i), :)'));

% Plot the data and the three lines
figure;
gscatter(X(:, 1), X(:, 2), y, 'rb', '+*'); % Plot data points
hold on;

% Plot the decision boundary (w1*x1 + w2*x2 + b = 0)
x_plot = linspace(0, 10, 100); % Adjusted x_plot range
y_plot_decision = (-w(1) * x_plot - b) / w(2);
plot(x_plot, y_plot_decision, 'k-', 'LineWidth', 2); % Black solid
line

% Plot the margin line (w1*x1 + w2*x2 + b = 1)
y_plot_margin_plus = (-w(1) * x_plot - b + 1) / w(2);
```

```
plot(x_plot, y_plot_margin_plus, 'g--', 'LineWidth', 1.5); % Green
dashed line

% Plot the margin line (w1*x1 + w2*x2 + b = -1)
y_plot_margin_minus = (-w(1) * x_plot - b - 1) / w(2);
plot(x_plot, y_plot_margin_minus, 'm--', 'LineWidth', 1.5); % Magenta
dashed line

% Set axis limits
xlim([0 10]); % Set x-axis limits from 0 to 10
ylim([0 10]); % Set y-axis limits from 0 to 10

title('SVM Decision Boundary and Margins (Adjusted)');
xlabel('X1');
ylabel('X2');
legend('Class 1', 'Class -1', 'Decision Boundary', 'Margin (+1)',
'Margin (-1)');
hold off;
```

- Results

The solution of the quadprog function gives the vector z, which contains the optimal alpha values. These alphas are then used to compute the weight vector w, and subsequently the bias term b. Together, w and b define the optimal separating hyperplane that classifies the data with the maximum margin shown in the plots below.

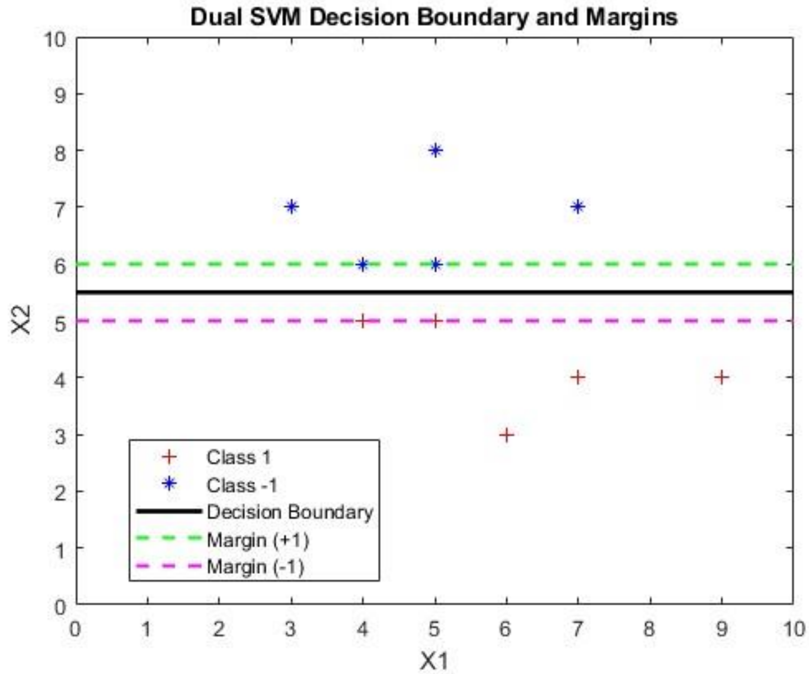

Figure 6: Results of the Dual Form SVM

III. Conclusion

In Conclusion, the perceptron method successfully performed weight updates whenever the output and target did not match. Additionally, both the direct and indirect SVM gave the optimal weights and hence the optimal classification line that correctly split the data into two classes. One can note that the same line is obtained both for the direct and indirect SVM and they also pass through the data (support vectors).