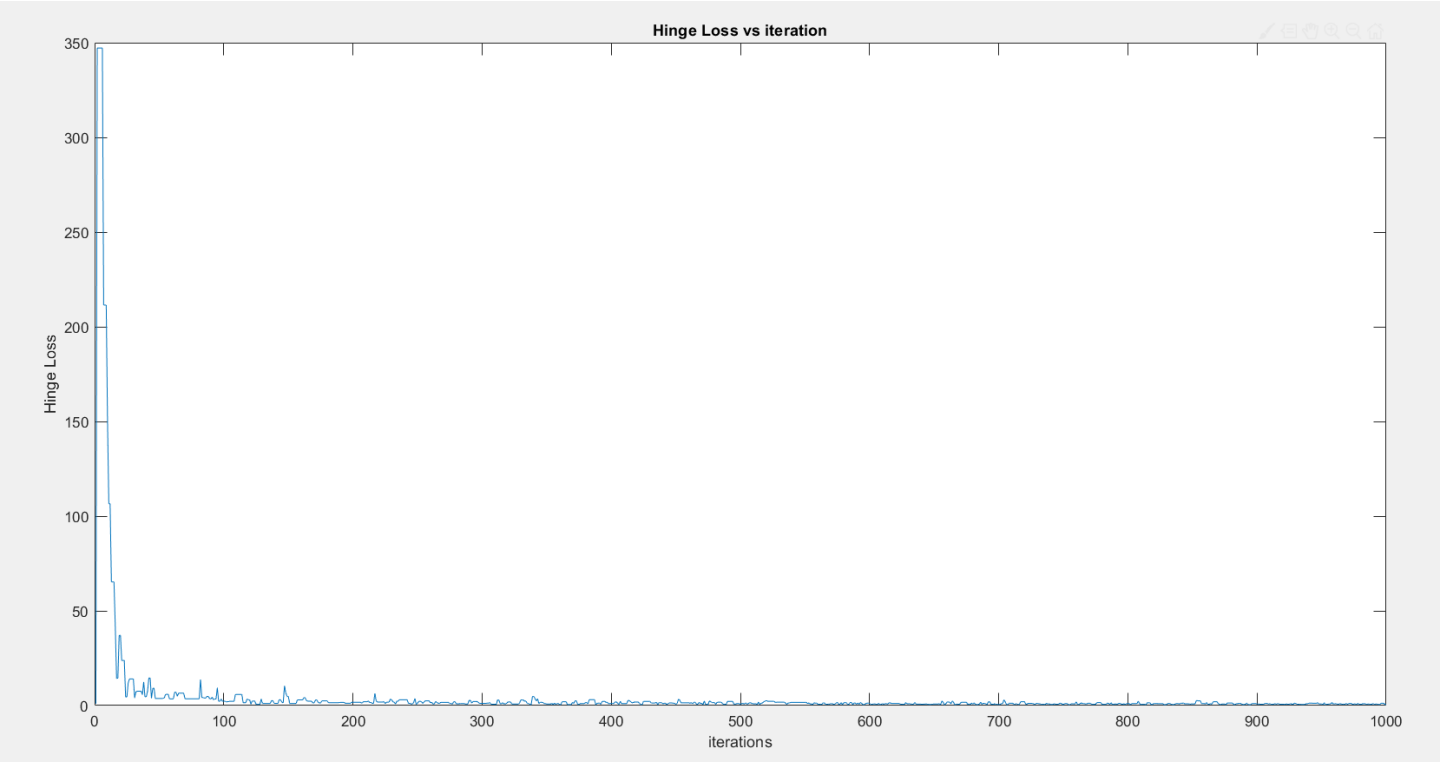
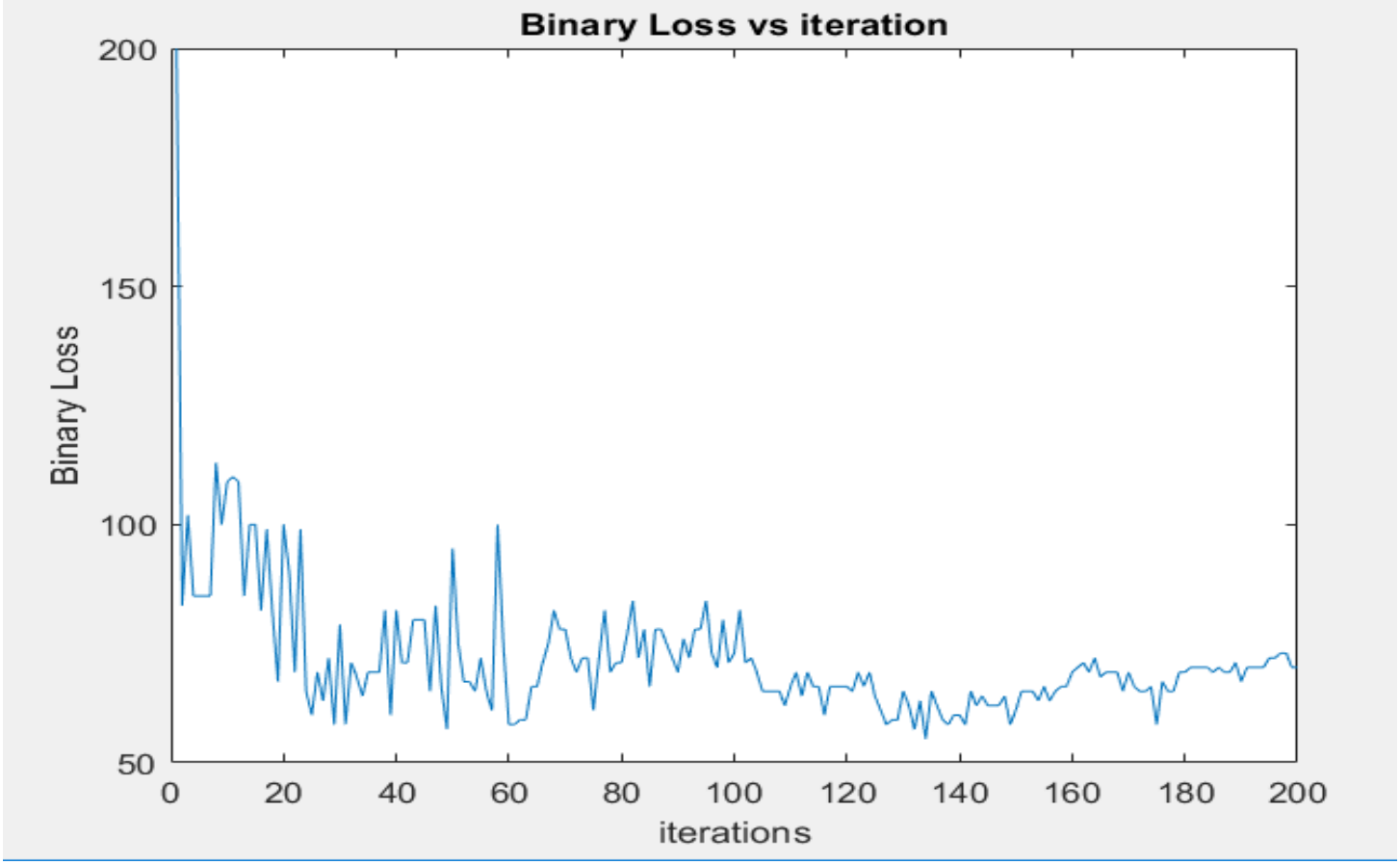
Question 3

Part 1:

The hinge loss plot that I got was as follows:



Binary Loss:



The code used is as follows:

```
clc;
close all;
clear all;

% Scanning input from the files
% Here are all the variables that we need to start
fid1=fopen('bg.txt');
s1=textscan(fid1,'%f,%f,%f,%f,%f');
fclose(fid1);
label1 = zeros(100, 5);
label0 = zeros(100, 5);
allFeatures = zeros(200, 5);
for k=1:5
input = s1{k};
label1(:,k) = input(1:100);
label0(:,k) = input(101:200);
allFeatures(:,k) = input;
end
allFeatures(:,6) = 1;
% Now here we start the soft svm code
% Set number of operations
T = 500;
u = (1:1:T);

% Set gamma here
gamma = 0.3;

% The weight vector is with 5 different variables
w = zeros(T,6);

% This is the subgradient
v = zeros(100, 6);

% Theta
theta = zeros(100, 6);

% Binay loss
binaryLoss = zeros(T, 1);
hingeLoss = zeros(T, 1);


% Now set Xi and Ti
Xi = allFeatures(1,:);
Ti = 1;
for j = 1:(T-1)
    % Current wait is w
    w(j,:) = (1/(j))*theta(j, :);
    wj = w(j,:);

    % Choosing a random variable
    index = randi([1 200]);

    % The random variable for X
    Xj = allFeatures(index,:);

    % Find if t is 1 or -1
    if (index > 100)
        t = -1;
    else
        t = 1;
    end
```

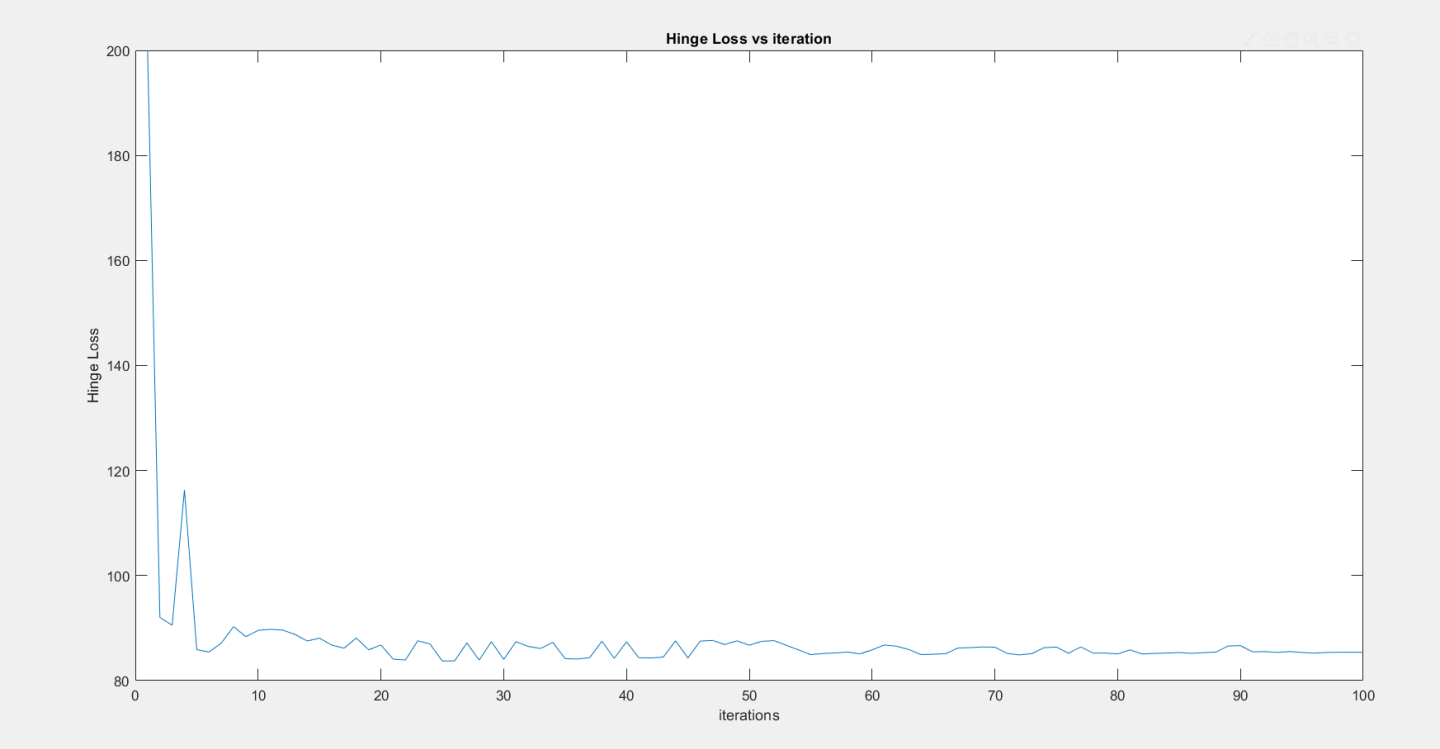
```
    % Dot product is t multiplied dot product of x and w
    dotProduct = t * dot(Xj, wj);
    % Lets set theta first
    if ((1-dotProduct) > 0)
        v(j,:) = -t * Xj;
        summation = 0;
        for p=1:j
            summation = summation + v(p,:);
        end
        theta(j,:) = -1 * summation;
    else
        v(j,:) = 0;
        theta(j,:) = -1 * sum(v);
    end



    % If X is misclassified then set theta accordingy
    if dotProduct < 0
        theta(j+1, :) = theta(j,:) + t * (Xj);
    else
        theta(j+1, :) = theta(j,:);
    end

    % Binary Loss finding
    for (m=1:200)
        X2 = allFeatures(m,:);
        if (m > 100)
            t2 = -1;
        else
            t2 = 1;
        end
        dotProduct2 = t2 * dot(X2, wj);
        if (dotProduct2 <= 0)
            binaryLoss(j) = binaryLoss(j) + 1;
        end
    end
    % Hinge Loss finding
    for (m=1:200)
        X2 = allFeatures(m,:);
        if (m > 100)
            t2 = -1;
        else
            t2 = 1;
        end
        dotProduct2 = t2 * dot(X2, wj);
        if (dotProduct2 > 0)
            hingeLoss(j) = hingeLoss(j) + 0;
        else
            hingeLoss(j) = hingeLoss(j) + (1 - (dotProduct2));
        end
    end
end
hingeLoss = hingeLoss/200;
plot(u, hingeLoss);
```

Part 2:

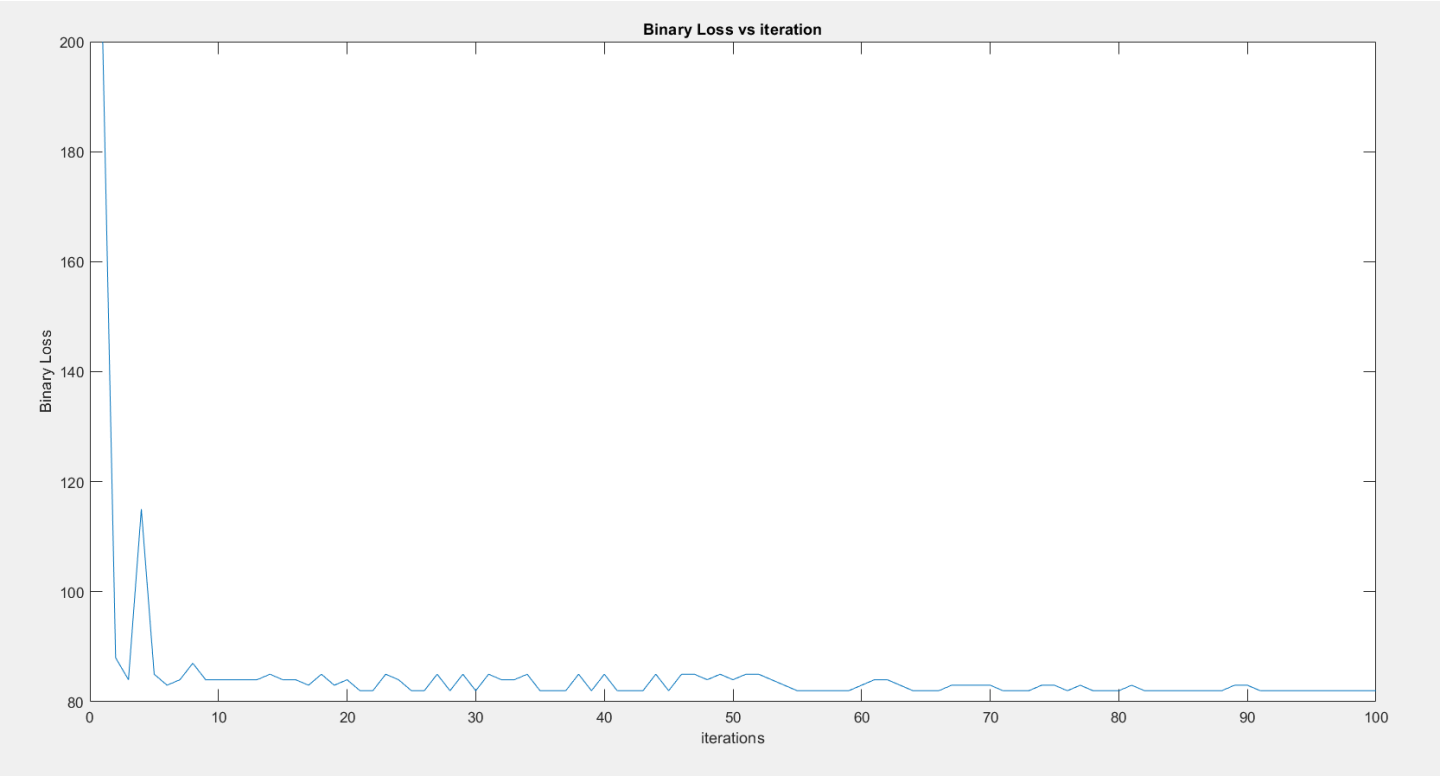
Now the three values of gamma I will show are 100, 1, and 0.01. I set the number of iteration to 100 to better see the difference between the different gamma values.

1- Gamma = 100:

Hinge loss:

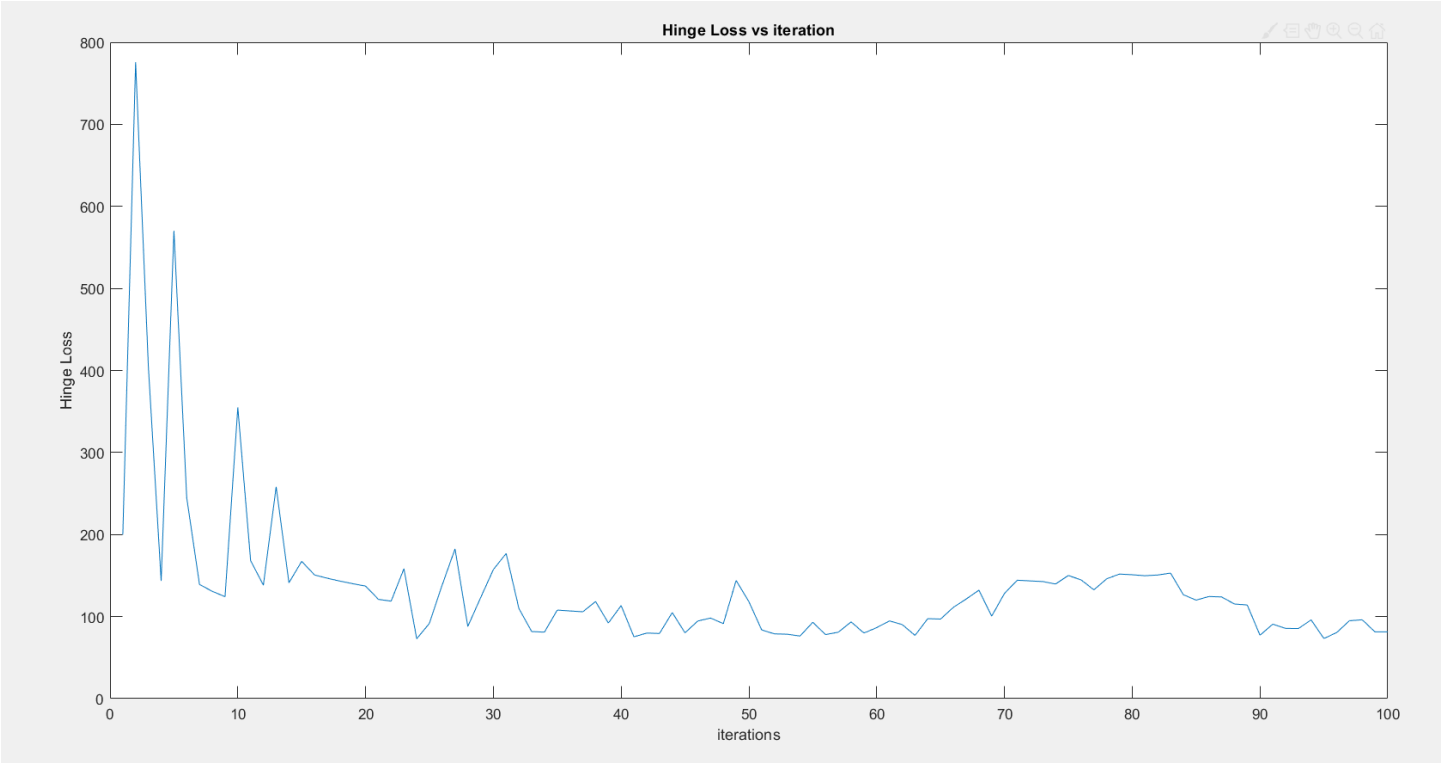


Binary loss:

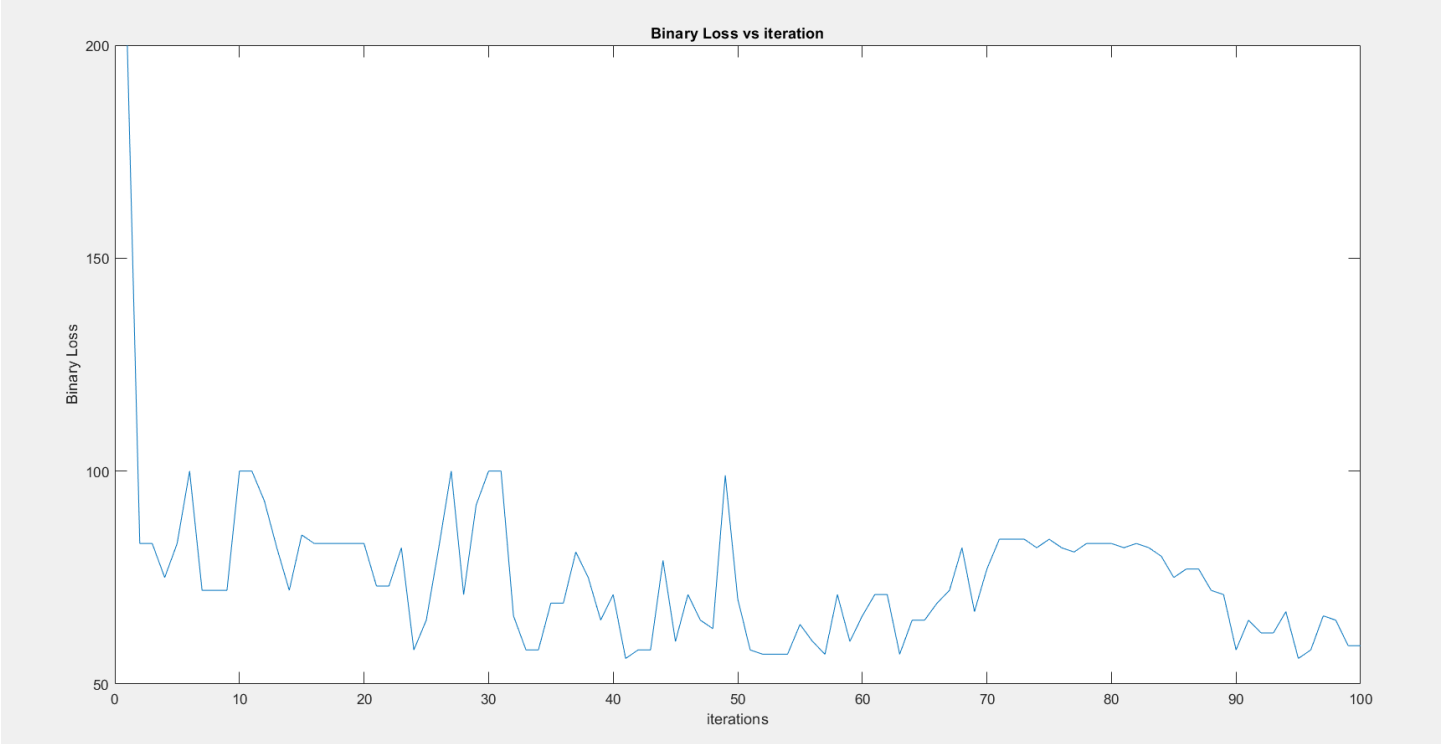


2- Gamma = 1

Hinge Loss:

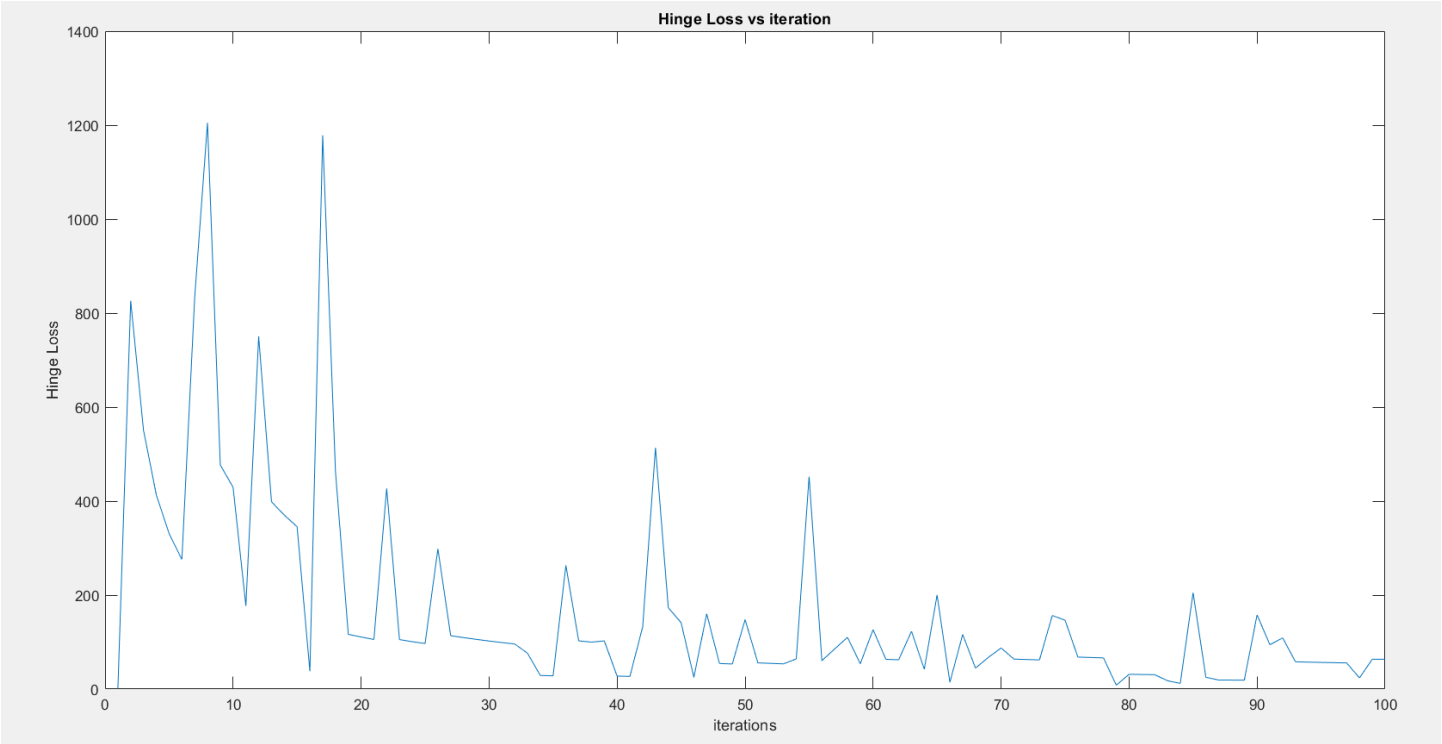


Binary Loss:

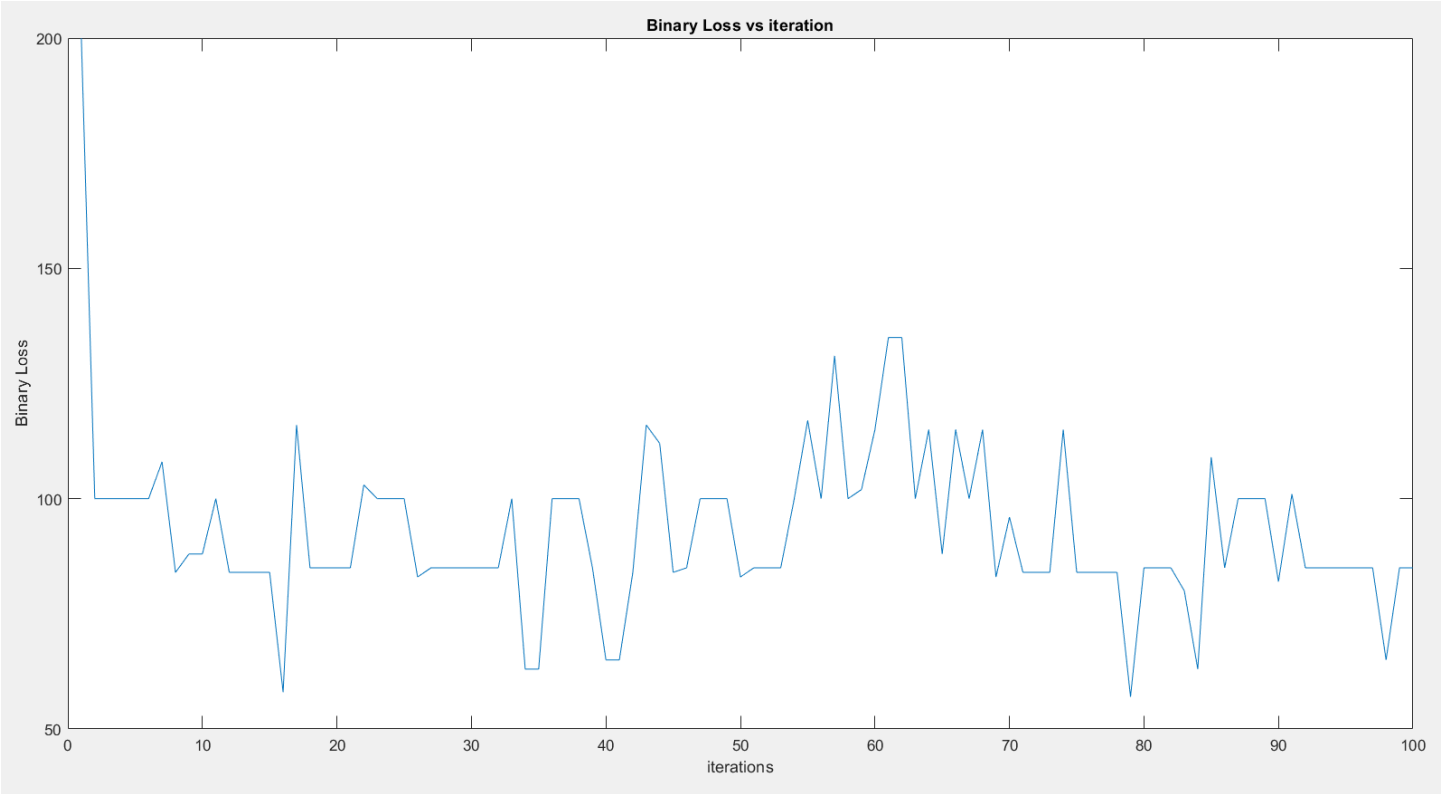


3- Gamma = 0.01

Hinge Loss:



Binary Loss:



Part 3:

Most of the plot were not completely monotone. The smaller the stepsize the more it approximated a monotone. The reason behind it, is if the stepsize kept getting smaller, there will be less big changes that will lead to big different in losses and hence becoming a better monotone approximation. We could increase and decrease the values of gamma to help tune the perfect step size and hence find better monotone functions. Using SGD, will help us find the value of w that has minimum loss, and the more monotone the equation the better and easier we could find the method.

Part 5:

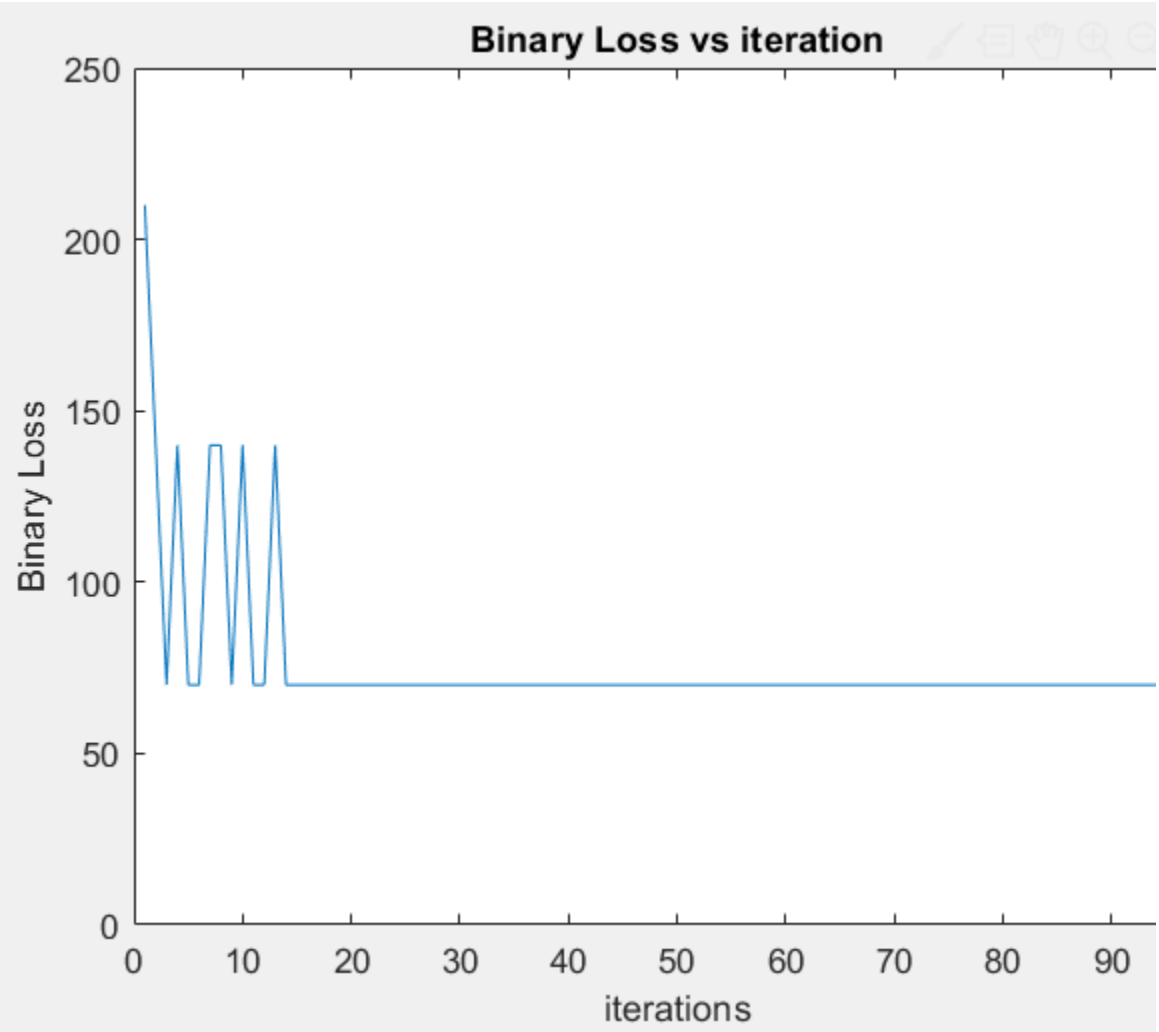
Using the following code:

```
clc;
close all;
clear all;

% Scanning input from the files
% Here are all the variables that we need to start
fid1=fopen('seeds_dataset.txt');
s1=textscan(fid1,'%f %f %f %f %f %f %f %d');
fclose(fid1);
allFeatures = zeros(210, 6);
for k=1:6
input = s1{k};
allFeatures(:,k) = input;
end
%allFeatures(:,7) = 1;
% Now here we start the soft svm code
% Set number of operations
T = 100;
u = (1:1:T);

% Set gamma here
gamma = 100;

% The weight vector is with 5 different variables
w = zeros(T,6);

% This is the subgradient
v = zeros(T, 6);

% Theta
theta = zeros(T, 6);

% Binay loss
binaryLoss = zeros(T, 1);
hingeLoss = zeros(T, 1);


% Now set Xi and Ti
Xi = allFeatures(1,:);
Ti = 1;
for j = 1:(T-1)
    % Current wait is w
    w(j,:) = (1/(gamma*j))*theta(j, :);
    wj = w(j,:);

    % Choosing a random variable
    index = randi([1 210]);

    % The random variable for X
```

```
    Xj = allFeatures(index,:);

    % Find if t is 1 or -1
    if (index >= 71 && index <141)
        t = 1;
    else
        t = -1;
    end

    % Dot product is t multiplied dot product of x and w
    dotProduct = t * dot(Xj, wj);
    % Lets set theta first
    if ((1-dotProduct) > 0)
        v(j,:) = -t * Xj;
        summation = 0;
        for p=1:j
            summation = summation + v(p,:);
        end
        theta(j,:) = -1 * summation;
    else
        v(j,:) = 0;
        summation = 0;
        for p=1:j
            summation = summation + v(p,:);
        end
        theta(j,:) = -1 * summation;
    end



    % If X is misclassified then set theta accordingy
    if dotProduct < 0
        theta(j+1, :) = theta(j,:) + t * (Xj);
    else
        theta(j+1, :) = theta(j,:);
    end

    % Binary Loss finding
    for (m=1:210)
        X2 = allFeatures(m,:);
        if (m >= 71 && m <141)
            t2 = 1;
        else
            t2 = -1;
        end
        m
        dotProduct2 = t2 * dot(X2, wj)
        if (dotProduct2 <= 0)
            binaryLoss(j) = binaryLoss(j) + 1;
        end
    end

end

figure();
plot(u, binaryLoss);
xlabel('iterations');
ylabel('Binary Loss');
title('Binary Loss vs iteration');
```

Explanation:
Here I used the same algorithm for the SGD for SVM.

The three binary losses that I found was 70 for each w.



Part 6:

In this part, I kept using the soft svm sgd implemented. The total number of misclassifications was 16. Therefore, empirical binary loss is 16/210.

This is the code I used:

```
clc;
close all;
clear all;

% Scanning input from the files
% Here are all the variables that we need to start
fid1=fopen('seeds_dataset.txt');
s1=textscan(fid1,'%f %f %f %f %f %f %f %d');
fclose(fid1);
allFeatures = zeros(210, 7);
for k=1:6
```

```
input = s1{k};
allFeatures(:,k) = input;
end
allFeatures(:,7) = 1;
stepSize = 0.001;
% Now here we start the soft svm code
% Set number of operations
T = 1000;
u = (1:1:T);

% Set gamma here
gamma = 100;

% The weight vector is with 5 different variables
w = zeros(T,7);

% This is the subgradient
v = zeros(T, 7);

% Theta
theta = zeros(T, 7);

% Binay loss
binaryLoss1 = zeros(T, 1);
hingeLoss = zeros(T, 1);


% Now set Xi and Ti
Xi = allFeatures(1,:);
Ti = 1;
for j = 1:(T-1)
    % Current wait is w
    wj = w(j,:);

    % Choosing a random variable
    index = randi([1 200]);

    % The random variable for X
    Xj = allFeatures(index,:);

    % Find if t is 1 or -1

    if (index > 70 && index < 141)
        t = 1;
    else
        t = -1;
    end


    % Dot product is t multiplied dot product of x and w
    dotProduct = t * dot(Xj, wj);
    % Lets set theta first
    if ((dotProduct) <= 0)
         w(j+1, :) = w(j,:) + t*Xj;
    else
        w(j+1, :) = w(j,:);
    end
end
w2 =wj;
```

```
% Set gamma here
gamma = 100;

% The weight vector is with 5 different variables
w = zeros(T,7);

% This is the subgradient
v = zeros(T, 7);

% Theta
theta = zeros(T, 7);

% Binay loss
binaryLoss = zeros(T, 1);
hingeLoss = zeros(T, 1);

for j = 1:(T-1)
    % Current wait is w
    wj = w(j,:);

    % Choosing a random variable
    index = randi([1 200]);

    % The random variable for X
    Xj = allFeatures(index,:);

    % Find if t is 1 or -1
    if (index >= 141)
        t = 1;
    else
        t = -1;
    end

    % Dot product is t multiplied dot product of x and w
    dotProduct = t * dot(Xj, wj);
    % Lets set theta first
    if ((dotProduct) <= 0)
         w(j+1, :) = w(j,:) + t*Xj;
    else
        w(j+1, :) = w(j,:);
    end

        % Binary Loss finding
    for (m=1:210)
        X2 = allFeatures(m,:);
        if (m >= 71 && m <141)
            t2 = 1;
        else
            t2 = -1;
        end
        m;
        dotProduct2 = t2 * dot(X2, wj);
        if (dotProduct2 <= 0)
            binaryLoss1(j) = binaryLoss1(j) + 1;
        end
    end
end
w3=wj;
```

```
% Set gamma here
gamma = 100;

% The weight vector is with 5 different variables
w = zeros(T,7);

% This is the subgradient
v = zeros(T, 7);

% Theta
theta = zeros(T, 7);

% Binay loss
binaryLoss = zeros(T, 1);
hingeLoss = zeros(T, 1);
for j = 1:(T-1)
    % Current wait is w
    wj = w(j,:);

    % Choosing a random variable
    index = randi([1 200]);

    % The random variable for X
    Xj = allFeatures(index,:);

    % Find if t is 1 or -1
    if (index < 71)
        t = 1;
    else
        t = -1;
    end

    % Dot product is t multiplied dot product of x and w
    dotProduct = t * dot(Xj, wj);
    % Lets set theta first
    if ((dotProduct) <= 0)
         w(j+1, :) = w(j,:) + t*Xj;
    else
        w(j+1, :) = w(j,:);
    end
end
w1 = wj;
w1 = w1 / norm(w1);
w2 = w2 / norm(w2);
w3 = w3 / norm(w3);

binaryLoss = 0;
for i=1:210
    Xj = allFeatures(i,:);
    dotProduct1 =  dot(w1, Xj);
    dotProduct2 =  dot(w2, Xj);
    dotProduct3 =  dot(w3,Xj);
    if (i<71)
        if ((dotProduct1 > dotProduct2) && (dotProduct1 > dotProduct2))
```

```
            ;
        else
            binaryLoss = binaryLoss + 1;
        end
    elseif (i>70 && i<141)
        if ((dotProduct2 > dotProduct1) && (dotProduct2 > dotProduct3))
            ;
        else
            binaryLoss = binaryLoss + 1;
        end
    else
        if ((dotProduct3 > dotProduct2) && (dotProduct3 > dotProduct1))
            ;
        else
            binaryLoss = binaryLoss + 1;
        end
    end
end
```