Problem 4.

a) 1. The test accuracy of the three different digits of C is shown below.

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
98.73%	99.32%	97.96%	97.47%	98.14%	97.60%	98.07%	98.28%	95.71%	96.31%

Table 1 Test Accuracy when C = 2

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
98.68%	99.22%	97.92%	97.42%	98.02%	97.49%	98.04%	98.21%	95.59%	96.31%

Table 2 Test Accuracy When C = 4

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
98.54%	99.13%	97.87%	97.48%	97.96%	97.36%	97.92%	98.08%	95.55%	96.36%

Table 3 Test Accuracy When C = 8

2. The number of support vectors for each digit when C equals to 2, 4, 8 is shown below.

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
464	505	1221	1422	900	1326	686	779	2093	1848

Table 4 The Number of Support Vectors of Each Digit When C = 2

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
455	486	1201	1416	880	1292	671	770	2097	1836

Table 5 The Number of Support Vectors of Each Digit When C = 4

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
436	455	1189	1400	869	1253	651	744	2077	1816

Table 6 The Number of Support Vectors of Each Digit When C = 8

As these six tables shown above, the digits 8 and 9 are with relative lowest test accuracy of all different values of C. Therefore, they have the two most support vectors to help classify.

3. The three support vectors of largest Lagrange multiplier on each side of the boundary when C equals to 2, 4, and 8 is shown below.

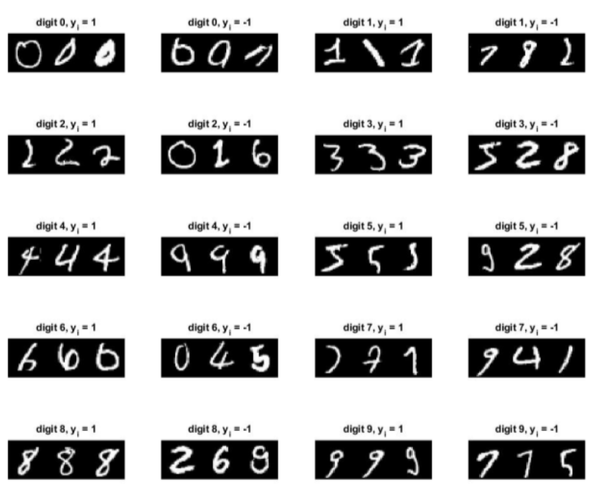


Figure 1 When C = 2

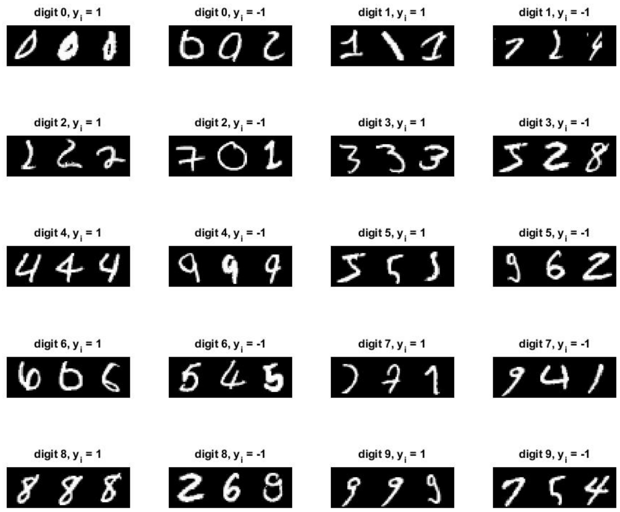


Figure 2 When C = 4

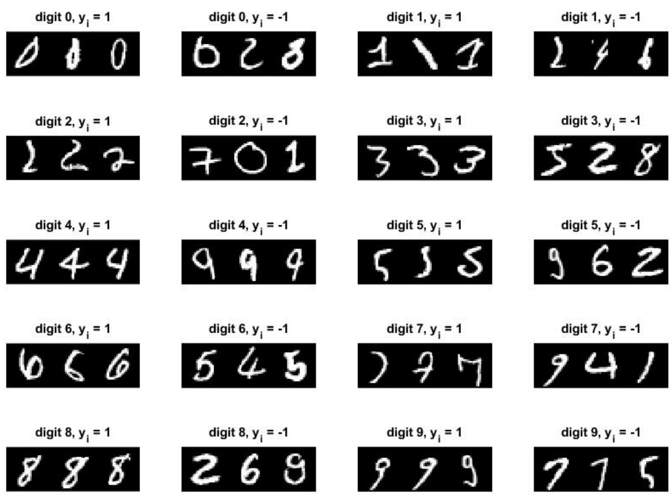


Figure 3 When C = 8

b) The cdf of the margin when C equals to 2, 4 and 8 is shown below.

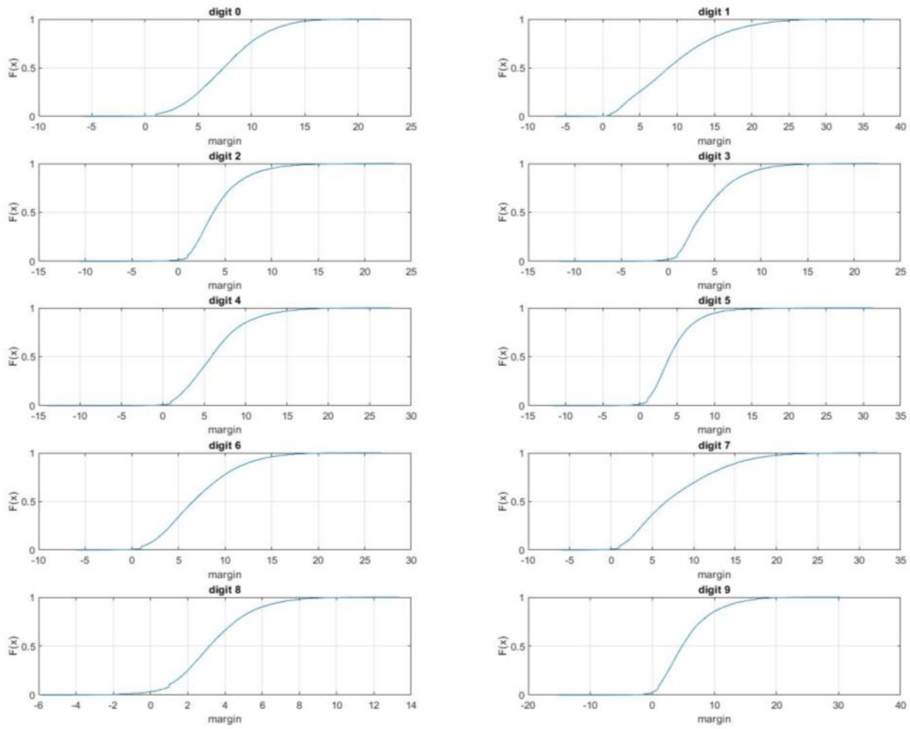


Figure 4 The cdf of Margin When C = 2

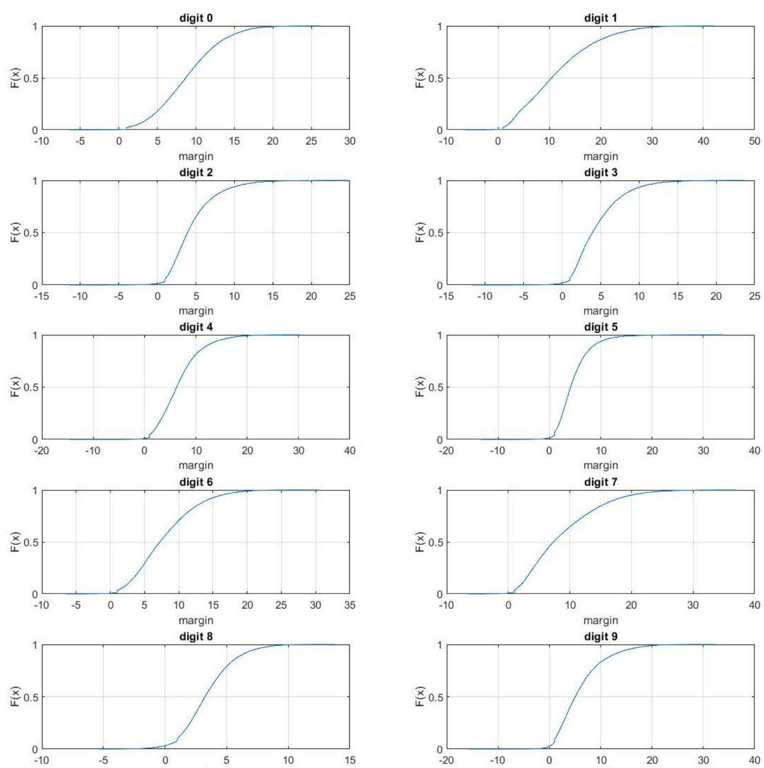


Figure 5 The cdf of Margin When C = 4

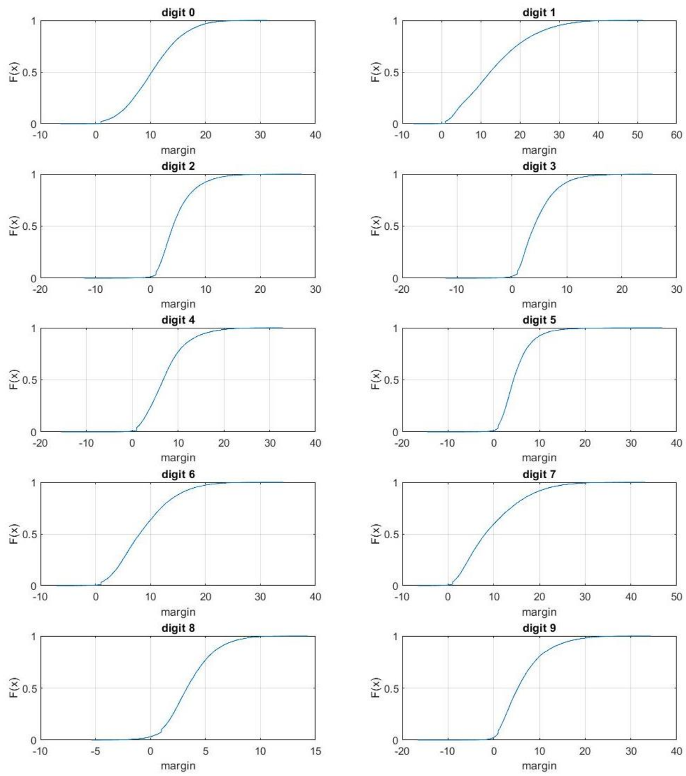


Figure 6 The cdf of Margin When C = 8

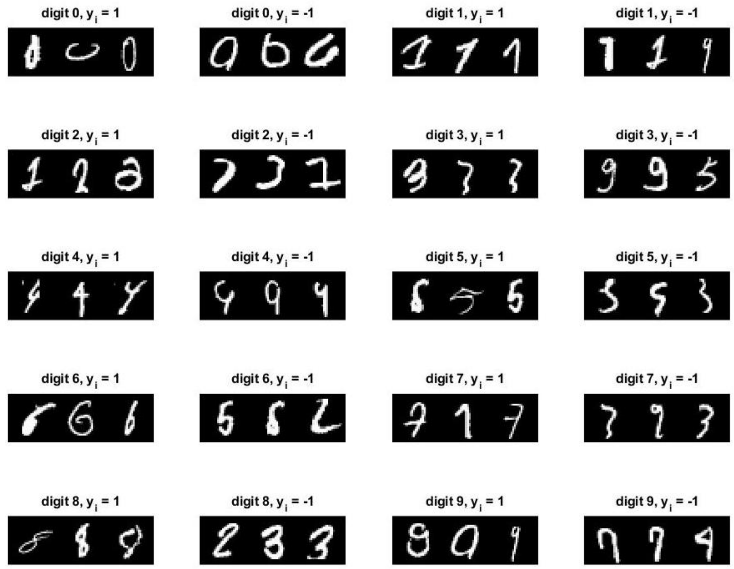
c) After running the script "grid.py", it gave us the value $C = 2, \gamma = 0.0625$.

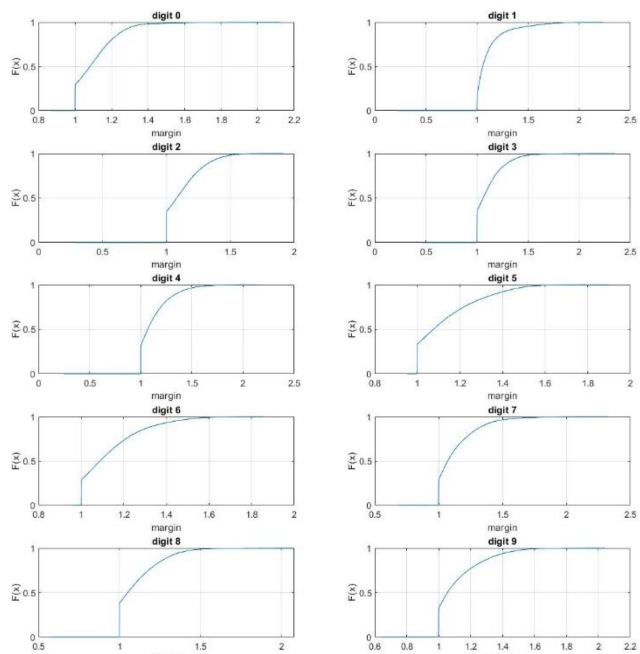
Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
99.53%	99.76%	98.88%	98.85%	99.15%	98.74%	99.30%	99.06%	98.36%	98.83%

Table 7 The Test Accuracy When C = 2, γ = 0.0625

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
5860	2419	6882	7123	6303	6682	5648	5734	7666	6488

Table 8 The Number of Support Vectors When C = 2, γ = 0.0625


digit 0, y_i = 1
digit 0, y_i = -1
digit 1, y_i = 1
digit 1, y_i = -1
digit 2, y_i = 1
digit 2, y_i = -1
digit 3, y_i = 1
digit 3, y_i = -1
digit 4, y_i = 1
digit 4, y_i = -1
digit 5, y_i = 1
digit 5, y_i = -1
digit 6, y_i = 1
digit 6, y_i = -1
digit 7, y_i = 1
digit 7, y_i = -1
digit 8, y_i = 1
digit 8, y_i = -1
digit 9, y_i = 1
digit 9, y_i = -1


digit 0
digit 1
digit 2
digit 3
digit 4
digit 5
digit 6
digit 7
digit 8
digit 9
F(x)
margin

```
clear;clc;close all;
num_train = 20000;
num_test= 10000;

[img_train, lbls_train] = readMNIST('MNISTdata/training set/train-images.idx3-
ubyte','MNISTdata/training set/train-labels.idx1-ubyte', num_train, 0);
[img_test, lbls_test] = readMNIST('MNISTdata/test set/t10k-images.idx3-
ubyte','MNISTdata/test set/t10k-labels.idx1-ubyte', num_test, 0);

target_train = labelReassign(lbls_train);
target_test =  labelReassign(lbls_test);
%% Part a & b
errors_test = zeros(3, 10);
numSV = zeros(3, 10);
pos = cell(3, 10);
neg = cell(3, 10);
C = [2,4,8];
for i = 1:3
    figure;
    for j = 1:10
        tic;
        model = svmtrain(target_train(:,j), img_train, ['-t 0 -c ', int2str(C(i))]);

        [pred_label, accuracy, dec_values] = svmpredict(target_test(:,j), img_test,
model);
        errors_test(i, j) = accuracy(1);
        numSV(i, j) = model.totalSV;
        [~, ind_max] = maxk(model.sv_coef, 3);
        [~, ind_min] = mink(model.sv_coef, 3);
        max3 = model.sv_indices(ind_max);
        min3 = model.sv_indices(ind_min);

        pos{i, j} = zeros(28, 28*3);
        neg{i, j} = zeros(28, 28*3);
        for k = 1:3
            pos{i, j}(:,k*28-27:k*28) = reshape(img_train(max3(k), :), [28, 28])';
            neg{i, j}(:,k*28-27:k*28) = reshape(img_train(min3(k), :), [28, 28])';
        end

        [pred, acc, dec] = svmpredict(target_train(:,j), img_train, model);
        subplot(5,2,j);
        cdfplot(dec .* target_train(:,j));
        xlabel('margin');
        title("digit " + (j-1));
        toc;
    end
end
%%
for i = 1:3
    figure(i);
    for j = 1:10
        pos_3 = pos{i,j};
        neg_3 = neg{i,j};

        subplot(5,4,2*j-1)
```

```
        imshow(pos_3);
        title("digit " +(j-1)+", y_{i} = 1");
        subplot(5,4,2*j)
        imshow(neg_3);
        title("digit " +(j-1)+", y_{i} = -1");
    end
end
%% Part c
errors_test2= zeros(1, 10);
numSV2 = zeros(1, 10);
pos2 = cell(1,10);
neg2 = cell(1,10);
figure;
for i=1:10
    model2 = svmtrain(target_train(:,i), img_train, '-c 2 -g 0.0625');

    [pred_label2, accuracy2, dec_values2] = svmpredict(target_test(:,i), img_test,
model2);
    errors_test2(i) = accuracy2(1);
    numSV2(i) = model2.totalSV;
    [~, ind_max] = maxk(model2.sv_coef, 3);
    [~, ind_min] = mink(model2.sv_coef, 3);
    max3 = model2.sv_indices(ind_max);
    min3 = model2.sv_indices(ind_min);

    pos2{i} = zeros(28, 28*3);
    neg2{i} = zeros(28, 28*3);
    for k = 1:3
        pos2{i}(:,k*28-27:k*28) = reshape(img_train(max3(k), :), [28, 28])';
        neg2{i}(:,k*28-27:k*28) = reshape(img_train(min3(k), :), [28, 28])';
    end

    [pred2, acc2, dec2] = svmpredict(target_train(:,i), img_train, model2);
    subplot(5,2,i);
    cdfplot(target_train(:,i).*dec2);
    xlabel('margin');
    title("digit "+(i-1));
end
%%
for j = 1:10
    pos_3 = pos2{j};
    neg_3 = neg2{j};

    subplot(5,4,2*j-1)
    imshow(pos_3);
    title("digit " +(j-1)+", y_{i} = 1");
    subplot(5,4,2*j)
    imshow(neg_3);
    title("digit " +(j-1)+", y_{i} = -1");
end
%%
function re_labels = labelReassign(labels)
    % assign 1 to the images of the specific class label and assign -1 to the rest of
classes
    features_num = size(unique(labels),1);
```

```
    re_labels = -1*ones(size(labels,1),features_num);
    for i = 1: size(labels,1)
        re_labels(i,labels(i)+1) = 1;
    end
end
```