Problem 5

(a)

Fig.1 shows the plots of the train and test error rates vs. iteration for binary classifiers. The error rates start from about 10% and decrease as iterations proceed.

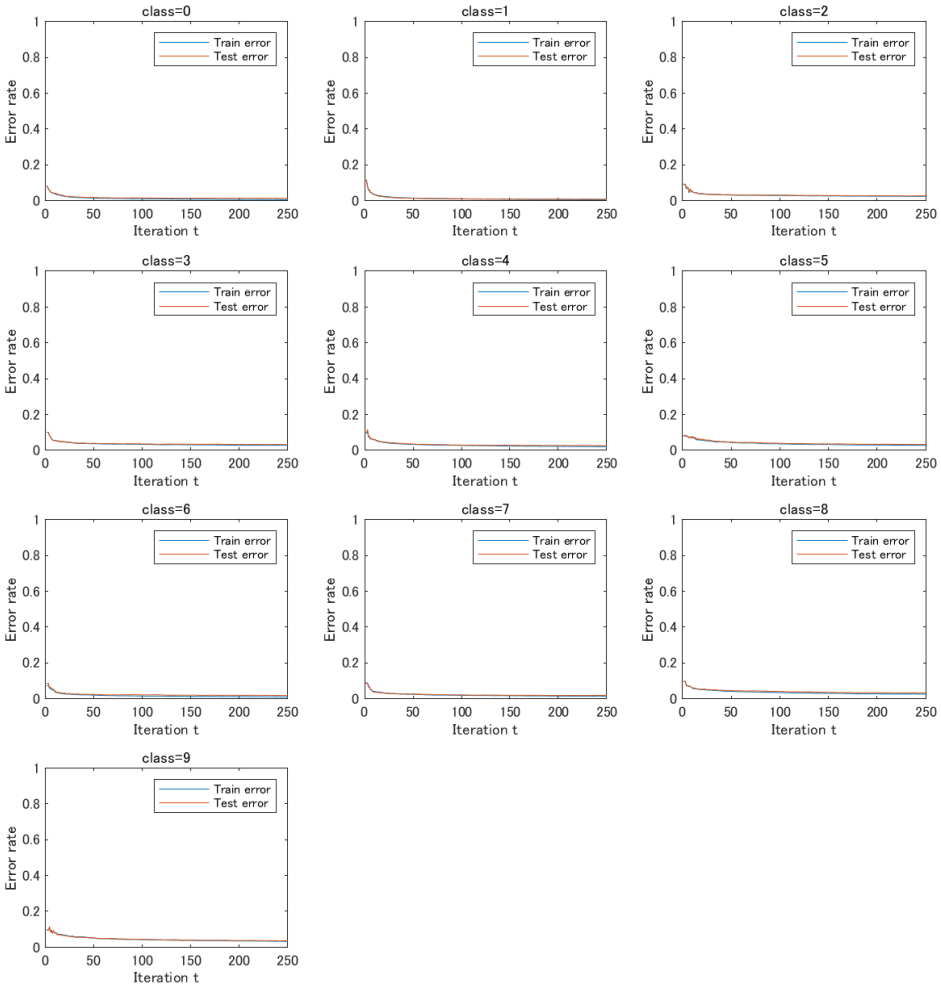

Fig.1 the train and test error rates vs. iteration for binary classifiers.

The Fig.2 shows the test error rate of the multi-class classifier. The final error rate is 0.0936.

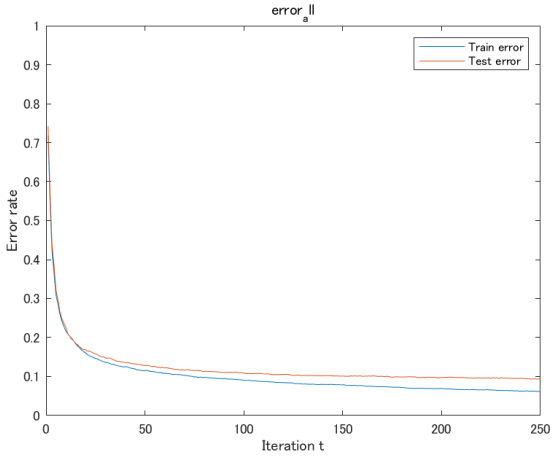

Fig.1 the train and test error rates vs. iteration for the multi-class classifier.

(b)

The Fig.3 shows the cumulative distribution function (CDF) for of the margins of all training samples after {5, 10, 50, 100, 250} iterations.

As iterations proceed, the CDF lines shift from left to right, and the range of the margins becomes greater. This means that the classifiers enforce the margins. Since the exponential loss function is used for this algorithm, the positive margins are also enforced.

The intersections of the DCF lines and the vertical axis are the error rates of the binary classifiers at the iteration. As iterations proceed, the intersection points go down, which means that the binary classifiers improve the accuracy for training data.

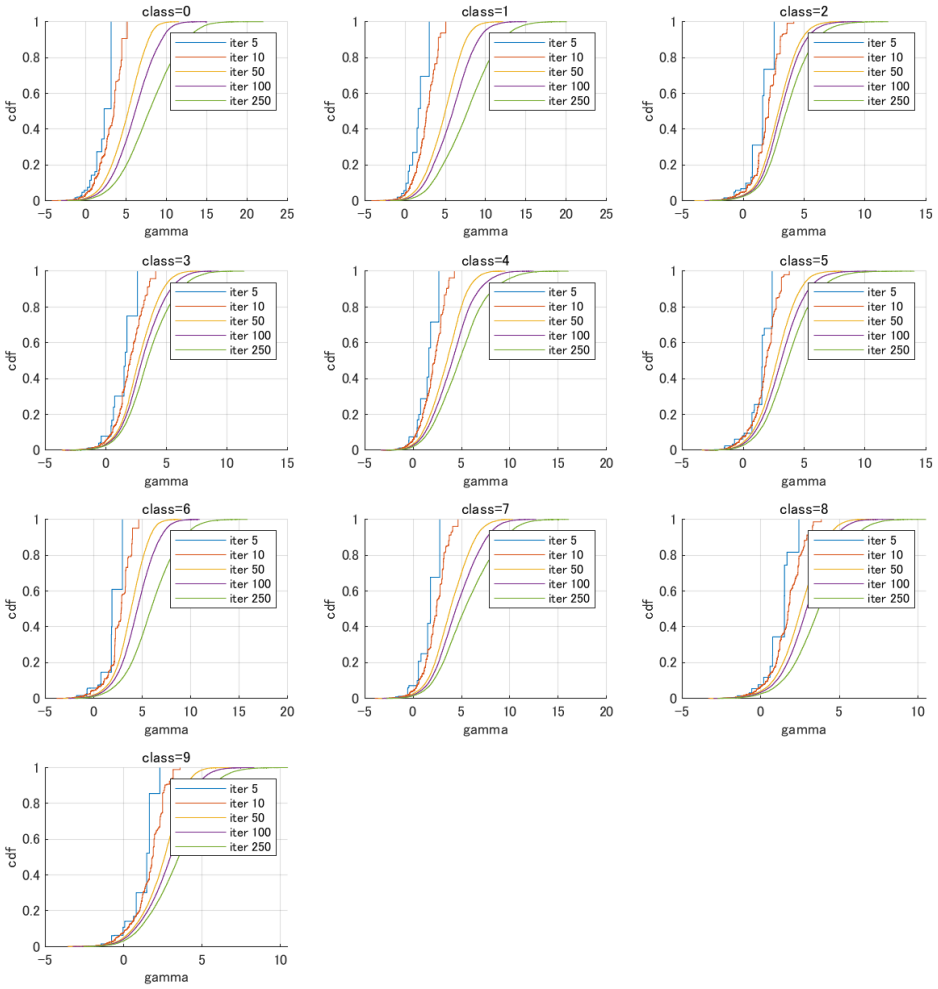

Fig.3 the cumulative distribution function for of the margins of all training samples after {5, 10, 50, 100, 250} iterations

(c)

The Fig.4 shows that the index of the example of largest weight for each boosting iteration. There are some difficult images to be classified.

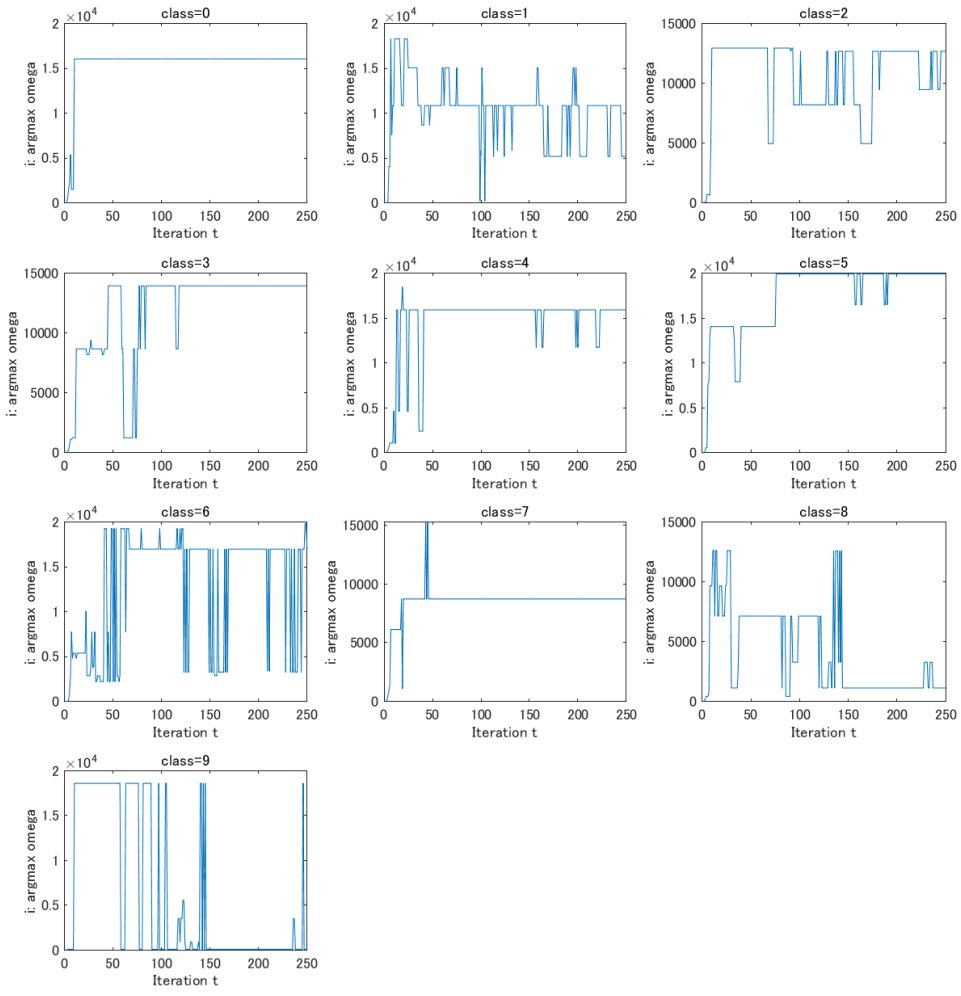

Fig.4 The index of the example of largest weight for each boosting iteration.

The Fig.5 shows the 3 examples that were most frequently selected, across boosting iterations, as the example of largest weight. According to the Fig.4, the left image of class 0 and class 7 are especially hard examples.

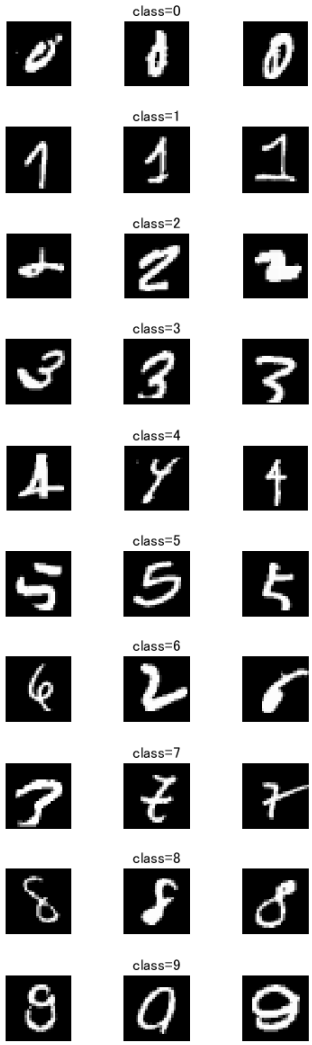

Fig.5 shows the 3 examples that were most frequently selected, across boosting iterations, as the example of largest weight.

(d)

Fig.6 show the visualized weak learners. A white pixel denotes a feature for regular weak learner, which outputs 1 for a value greater than its threshold. A black pixel represents a feature for a twin weak learner, which outputs -1 for a value greater than its threshold. Therefore, these images have vague shapes of the corresponding digits.

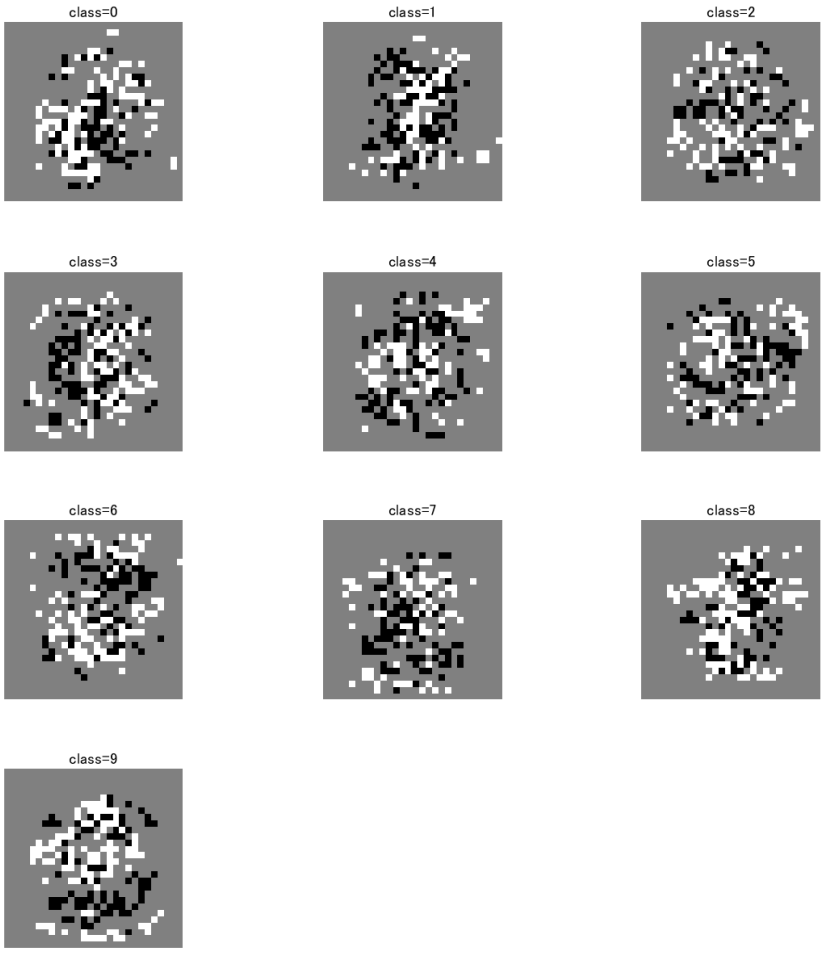

Fig.6 the visualized weak learners.

Code:

```
% clc;
clear all;
trainImgFile = 'training set¥train-images-idx3-ubyte';
trainLblFile = 'training set¥train-labels-idx1-ubyte';
[trainImgs, trainLabels] = readMNIST(trainImgFile,trainLblFile,20000,0);

testImgFile = 'test set¥t10k-images-idx3-ubyte';
testLblFile = 'test set¥t10k-labels-idx1-ubyte';
[testImgs, testLabels] = readMNIST(testImgFile,testLblFile,10000,0);

n = size(trainImgs,1);
t_train = - ones(n, 10);
for i = 1:n
    t_train(i,trainLabels(i)+1) = 1;
end
x_train = [trainImgs ones(n,1)];
x_test = [testImgs ones(size(testImgs,1),1)];

gt_train = zeros(n,10);
gt_test  = zeros(size(testImgs,1),10);
T = 250;
% T = 50;
alphaj = zeros(n,10);
alphathresh = zeros(n,10);
alphasgn = zeros(n,10);
alphax_train = zeros(n,1);
alphax_test  = zeros(size(testImgs,1),1);
err_train = zeros(T,1);
err_test  = zeros(T,1);
err_c_train1 = zeros(T,10);
err_c_test1  = zeros(T,10);
err_c_train2 = zeros(T,10);
err_c_test2  = zeros(T,10);
```

```
i_star = zeros(T,10);
times = [5 10 50 100 250];
gammas = zeros(n,10,5);
for t = 1:T
    for c = 1:10
        y = t_train(:,c);
        gamma = y.*gt_train(:,c);
        omega = exp(-gamma);
        [~, i_star(t,c)] = max(omega);
        for j = 1:5
            if t == times(j)
                gammas(:,c,j) = gamma;
            end
        end
        [alphaj(t,c), alphathresh(t,c), alphasgn(t,c), epsiron] = search_alpha(omega,
x_train, y);
        if epsiron == 0
            continue;
        end
        wt = 0.5 * log((1-epsiron)/epsiron);
        alphax_train = u(x_train, alphaj(t,c), alphathresh(t,c), alphasgn(t,c));
        alphax_test  = u(x_test,  alphaj(t,c), alphathresh(t,c), alphasgn(t,c));
        gt_train(:,c) = gt_train(:,c) + wt*alphax_train;
        gt_test(:,c)  = gt_test(:,c)  + wt*alphax_test;

        err_c_train1(t,c) = sum(xor(gt_train(:,c)>0, trainLabels==c-1)) /
size(trainLabels,1);
        err_c_test1(t,c)  = sum(xor(gt_test(:,c)>0, testLabels==c-1)) /
size(testLabels,1);
    end
    [~, predicted_train] = max(gt_train, [], 2);
    [~, predicted_test] = max(gt_test, [], 2);
    err_train(t) = 1 - sum(predicted_train-1==trainLabels) / numel(predicted_train);
    err_test(t) = 1 - sum(predicted_test-1==testLabels) / numel(predicted_test);
    for c = 1:10
        err_c_train2(t,c) = sum(xor(predicted_train-1==c-1, trainLabels==c-1)) /
```

```
numel(predicted_train);
        err_c_test2(t,c) = sum(xor(predicted_test-1==c-1, testLabels==c-1)) /
numel(predicted_test);
    end
end

show_error_all(err_train, err_test, T);
show_error_each(err_c_train1, err_c_test1, T, 'error_each1.png');
show_error_each(err_c_train2, err_c_test2, T, 'error_each2.png');
show_gammas(gammas);
show_i_star(i_star, T);
a = weak_learner_img(alphaj, alphasgn, T);

err_test(T)

figure('Renderer', 'painters', 'Position', [10 10 300 900]);
for c=1:10

    [ii,bin] = hist(i_star(:,c), unique(i_star(:,c)));
    [~, iii] = sort(-ii);
    i_star1 = bin(iii(1));
    i_star2 = bin(iii(2));
    i_star3 = bin(iii(3));

    ax = subplot(10,3,3*(c-1)+1);
    imshow(reshape(trainImgs(i_star1,:), [28 28])');
    ax = subplot(10,3,3*(c-1)+2);
    imshow(reshape(trainImgs(i_star2,:), [28 28])');
    title(['class=' int2str(c-1)]);
    ax = subplot(10,3,3*(c-1)+3);
    imshow(reshape(trainImgs(i_star3,:), [28 28])');
end
saveas(gcf, 'i_star_img.png');

save('all.mat');
```

```
function [bestj,bestt,bestsgn, epsiron] = search_alpha(omega, x_train, y)
    N = 50;
    %N = 5;
    bestsum = 0;
    for j = 1:748
        for i = 0:N
            thresh  = i/N;
            r = u(x_train, j, thresh, 1);
            ssum = sum(y.*r.*omega);
            twinsum = -ssum;
            if ssum > bestsum
                bestj = j;
                bestt = thresh;
                bestsgn = 1;
                bestsum = ssum;
            end
            if twinsum > bestsum
                bestj = j;
                bestt = thresh;
                bestsgn = -1;
                bestsum = twinsum;
            end
        end
    end
    r = u(x_train, bestj, bestt, bestsgn);
    epsiron = sum(omega(r~=y))/sum(omega);
end

function res = u(x, j, thresh, sgn)
    xj = x(:,j);
    res = xj >= thresh;
    res = res*2-1;
    res = res*sgn;
end
```

```
function show_error_all(trainError, testError, T)
    figure;
    plot(1:T, trainError, 1:T, testError);
    legend('Train error', 'Test error');
    title('error_all');
    xlabel('Iteration t');
    ylabel('Error rate');
    ylim([0 1]);
    saveas(gcf, 'error_all.png');
 end

function show_error_each(err_c_train, err_c_test, T, filename)
    figure('Renderer', 'painters', 'Position', [10 10 900 900]);
    for c = 1:10
        discription = ['class=' int2str(c-1)];
        ax = subplot(4,3,c);
        plot(1:T, err_c_train(:,c), 1:T, err_c_test(:,c));
        legend('Train error', 'Test error');
        title(discription);
        xlabel('Iteration t');
        ylabel('Error rate');
        ylim([0 1]);
    end
    saveas(gcf, filename);
end

function show_gammas(gammas)
    figure('Renderer', 'painters', 'Position', [10 10 900 900]);
    for c = 1:10
        ax = subplot(4,3,c);
        hold on;
        for j = 1:5
            gamma = gammas(:,c,j);
            cdfplot(gamma);
        end
        legend('iter 5', 'iter 10', 'iter 50', 'iter 100', 'iter 250');
```

```
        title(['class=', int2str(c-1)]);
        xlabel('gamma');
        ylabel('cdf');
        hold off;
    end
    saveas(gcf, 'gammas.png');
end

function show_i_star(i_star, T)
    figure('Renderer', 'painters', 'Position', [10 10 900 900]);
    for c = 1:10
        ax = subplot(4,3,c);
        plot(1:T, i_star(:,c));
        title(['class=' int2str(c-1)]);
        xlabel('Iteration t');
        ylabel('i: argmax omega');
    end
    saveas(gcf, 'i_star.png');
end

function a = weak_learner_img(alphaj, alphasgn, T)
    figure('Renderer', 'painters', 'Position', [10 10 900 900]);
    a = 128*ones(10, 28*28);
    for c = 1:10
        for t = 1:T
            a(c, alphaj(t,c)) = 255*(alphasgn(t,c)+1)/2;
        end
        a_img = reshape(a(c,:), [28 28])';
        ax = subplot(4,3,c);
        imshow(a_img/255);
        title(['class=' int2str(c-1)]);
    end
    saveas(gcf, 'weak_learner_img.png');
end
```