D = 784;

N = 400;

x = table2array(MNIST3);

% 2(a)

% mean vector of x

x\_bar = sum(x)/400;

% data covariance

S = zeros(784, 784);

t = linspace(1,100,100);

for i = 1:N

temp = (x(i,:) - x\_bar);

temp2 = transpose(temp)\*temp;

S = S + temp2;

end

S = S/N;

% eigenvalue

eigen = eig(S);

eigen = sort(eigen, 'descend');

eigen = eigen(1:100);

figure;

scatter(t, eigen);

% 2(b)

% eigenvector

[V, Diag] = eig(S); % V is the corresponding right eigenvectors

V = V(:, D-3:D); % get the first 4 eigenvectors

% rescale

for i = 1:4

max\_pixel = max(V(:, i));

V(:,i) = 255/max\_pixel \* V(:,i);

end

% visualization

figure;

for i = 1:4

img = zeros(28,28);

for j = 1:784

a = (j-1 - mod(j-1, 28))/28 + 1;

b = mod(j-1, 28)+1;

img(a,b) = V(j, i);

end

subplot(2,2,i);

imshow(transpose(img));

end

% 2(c)

figure;

% show first image

img = zeros(28,28);

for j = 1:D

a = (j-1 - mod(j-1, 28))/28 + 1;

b = mod(j-1, 28)+1;

img(a,b) = x(1, j);

end

subplot(3,2,1);

imshow(transpose(img));

title('Original');

% show compressed image

M = [0 1 10 50 250];

[V, Diag] = eig(S); % V is the corresponding right eigenvectors

for m = 1:5

c\_img = zeros(28,28);

x\_tilda = zeros(1,D);

for i = 1:M(m)

eigen\_vec = V(:,785-i); %784\*1

temp1 = (x(1,:)\*eigen\_vec-x\_bar\*eigen\_vec) \* eigen\_vec;

x\_tilda = x\_tilda + transpose(temp1);

end

x\_tilda = x\_tilda + x\_bar;

for j = 1:D

a = (j-1 - mod(j-1, 28))/28 + 1;

b = mod(j-1, 28)+1;

c\_img(a,b) = x\_tilda(1, j);

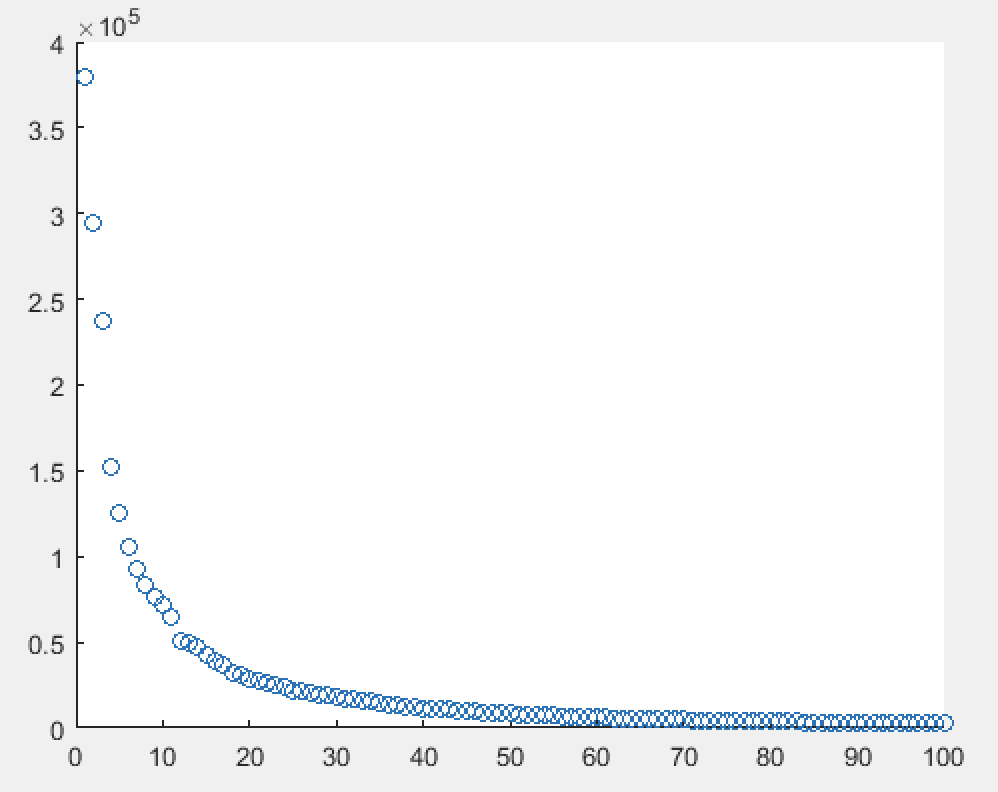
end

subplot(3,2,m+1);

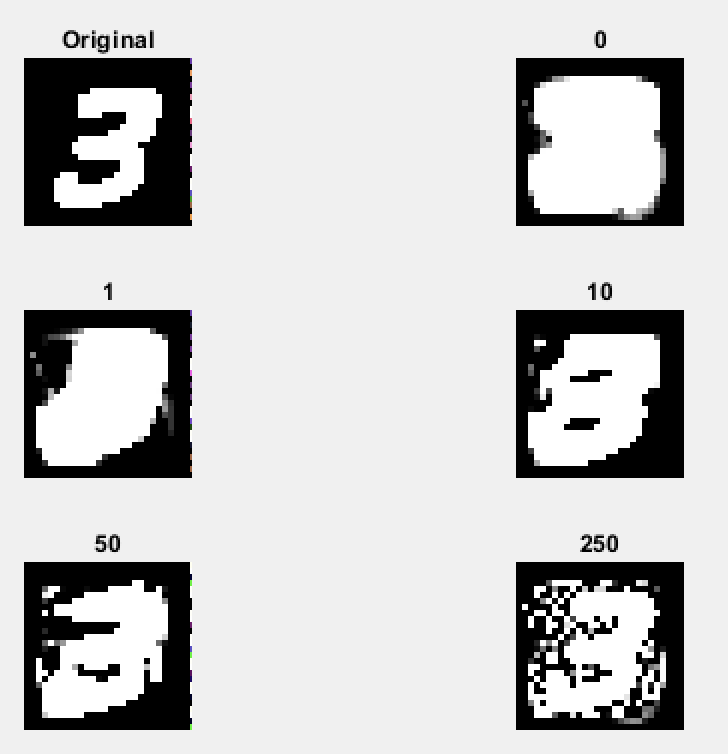
imshow(transpose(c\_img));

title(M(m));

end







% 6(a)

data = table2array(AdaBoostdata);

x1 = data(:,1);

x2 = data(:,2);

y = data(:,3);

figure;

hold on;

for i = 1:size(y)

if (y(i) == 1)

scatter(x1(i),x2(i),'b');

else

scatter(x1(i),x2(i),'r');

end

end

hold off;

% 6(b)

K = 3;

N = length(y);

f = zeros(1,K);

alph = zeros(1,K);

% Initialize uniform importance

dk = ones(4,N)/N;

d(1:N) = (1/N) \* ones(N,1);

for k = 1:K

if k == 1 % given optimal s and i

i = 1; s = -1;

elseif k == 2

i = 1; s = -1;

else

i = 2; s = 1;

end

err = zeros(1,N);

% Find optimal t

for t = 0:N-1

for n = 1:N

y\_prd(n) = sign((data(n,i)-t)\*s); % Make predictions on training data

if y\_prd(n)~= y(n) % Error

err(t+1) = err(t+1) + d(n); % Compute weighted training error

end

end

end

[min\_err, min\_ind] = min(err);

f(k) = min\_ind-1; % t

alph(k) = 1/2\*log((1-min\_err)/min\_err); % Compute adaptive parameter

% Re-weight examples and normalize

for n = 1:N

y\_prd(n) = sign((data(n,i)-f(k))\*s);

d(n) = d(n)\*exp(-alph(k)\*y(n)\*y\_prd(n)); % update weight on misclassified point

end

% normalize

d = d / sum(d);

dk(k+1,:) = d;

end

% Final classifer

% y = sign(0.4236\*(3-x1)+0.6496\*(7-x1)+0.9229\*(x2-5))

% Classifier Accuracy

e = 0;

y\_p = zeros(1,10);

for n = 1:N

y\_p(n) = sign(0.4236\*sign(3-x1(n))+0.6496\*sign(7-x1(n))+0.9229\*sign(x2(n)-5));

if y\_p(n) ~= y(n)

e = e + 0.1;

end

end

% Plot for k=1,2,3

figure;

subplot(3,1,1);

hold on;

for i = 1:length(y)

if (y(i) == 1)

scatter(x1(i),x2(i),dk(1,i)\*1000,'b');

else

scatter(x1(i),x2(i),dk(1,i)\*1000,'r');

end

end

line([3,3],[0,7]);

hold off;

subplot(3,1,2);

hold on;

for i = 1:length(y)

if (y(i) == 1)

scatter(x1(i),x2(i),dk(2,i)\*1000,'b');

else

scatter(x1(i),x2(i),dk(2,i)\*1000,'r');

end

end

line([7,7],[0,7]);

hold off;

subplot(3,1,3);

hold on;

for i = 1:length(y)

if (y(i) == 1)

scatter(x1(i),x2(i),dk(3,i)\*1000,'b');

else

scatter(x1(i),x2(i),dk(3,i)\*1000,'r');

end

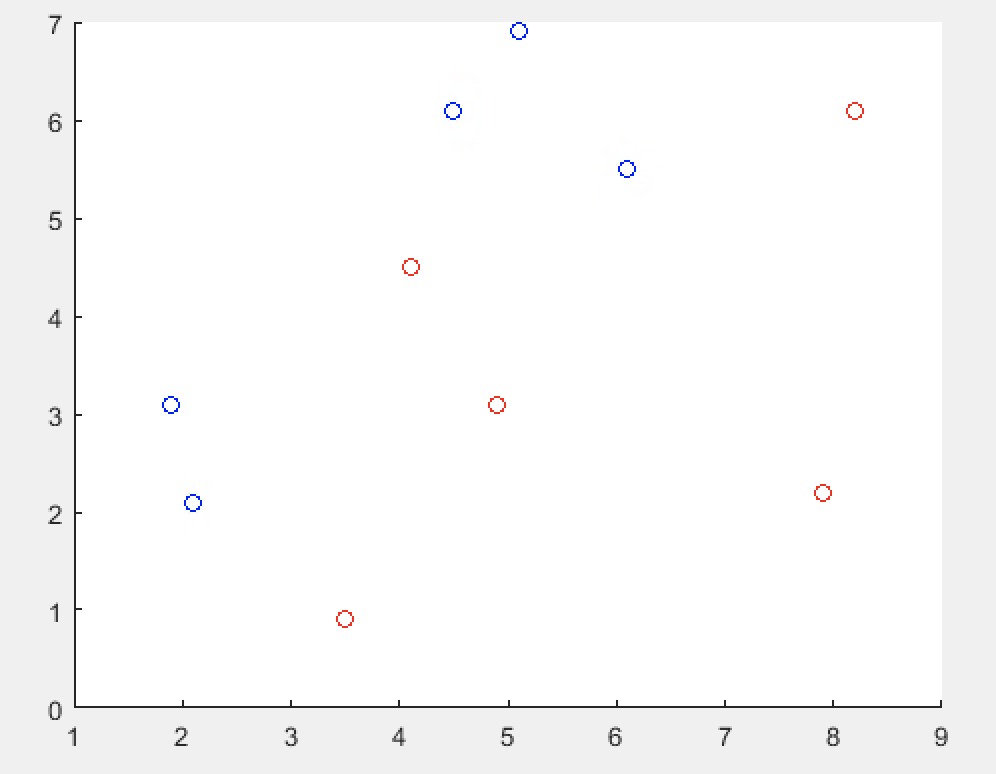
end

line([0,10],[5,5]);

hold off;

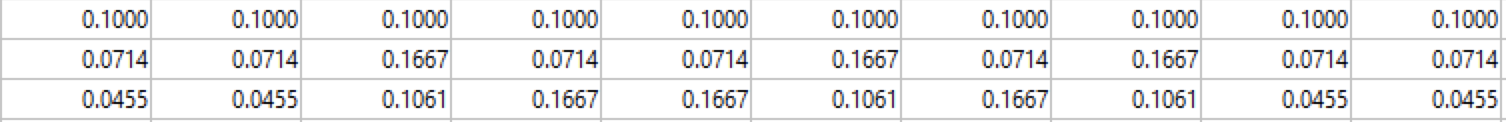
6(a)

The data points are not linearly separable. A single layer decision tree cannot classify them correctly.



6(b)

d:



t = [3 7 5]

alpha = [0.423648930193602 0.649641492065130 0.922913345249165]

y = sign(0.4236\*sign(3-x1(n))+0.6496\*sign(7-x1(n))+0.9229\*sign(x2(n)-5))

The training accuracy is 100%.

