COMS 4721 Machine Learning for Data Science

HW#3

UNI: sl3763

Name: Sung-Yen Liu

Problem1

Code:	

<pre>function x = sampleDiscrete(n, k)
 	
if (sum(k)~=1)
        knorm = k/sum(k);
else
        knorm = k;
end
 	
kcum = cumsum([0 knorm]);
rd = rand(1,n);
[t1, t2] = histc(rd,kcum);
ss = 1:length(k);
x=ss(t2);
 	
end</pre>
	<pre>w = [0.1 0.2 0.3 0.4];
	
	aa = sampleDiscrete(100,w);
	hist(aa,4), title(['n=100'])
	aa = sampleDiscrete(200,w);
	hist(aa,4), title(['n=200'])
	aa = sampleDiscrete(300,w);
	hist(aa,4), title(['n=300'])
	aa = sampleDiscrete(400,w);
	hist(aa,4), title(['n=400'])
	aa = sampleDiscrete(500,w);
	hist(aa,4), title(['n=500'])</pre>

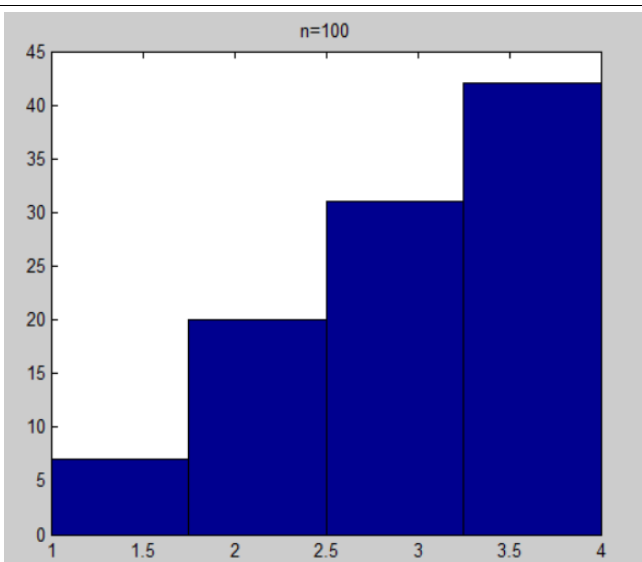
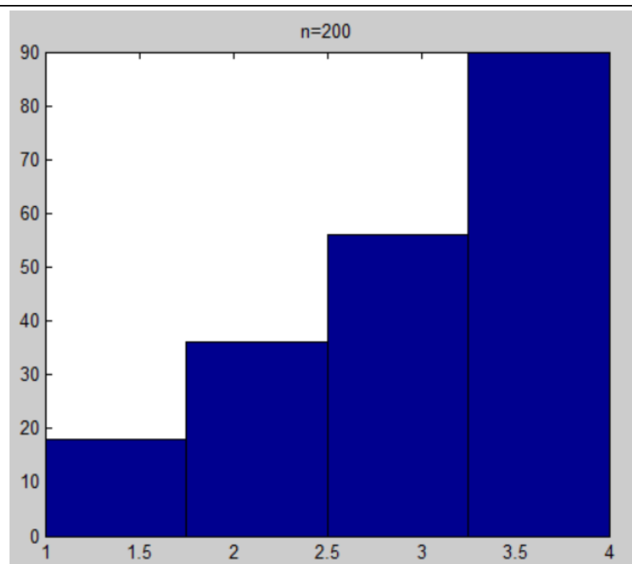
This function utilizes CDF function and random number generator.

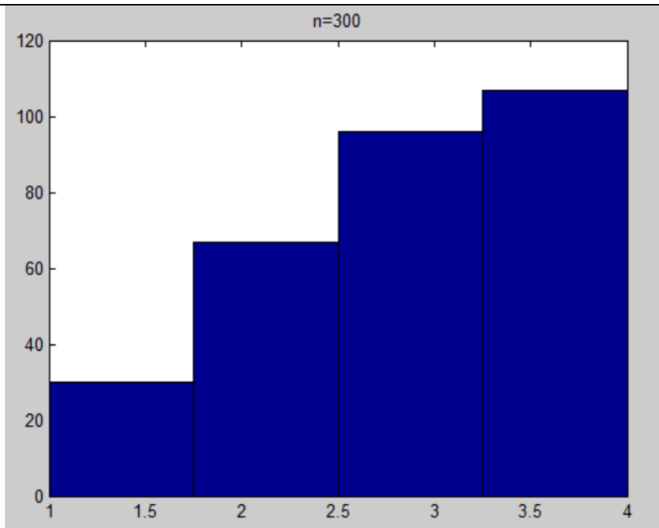
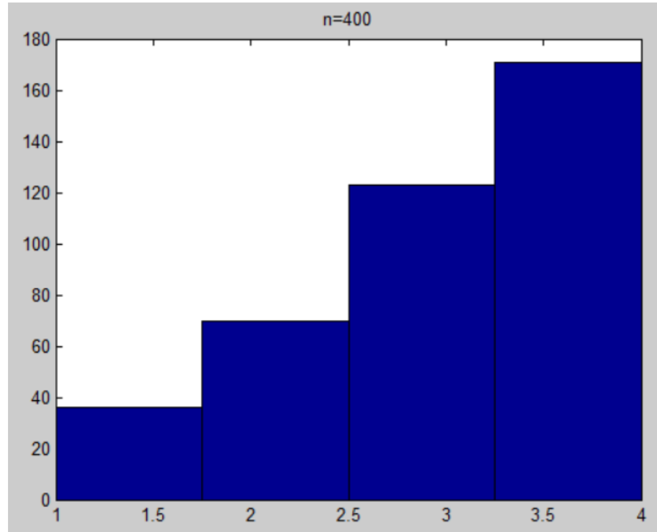
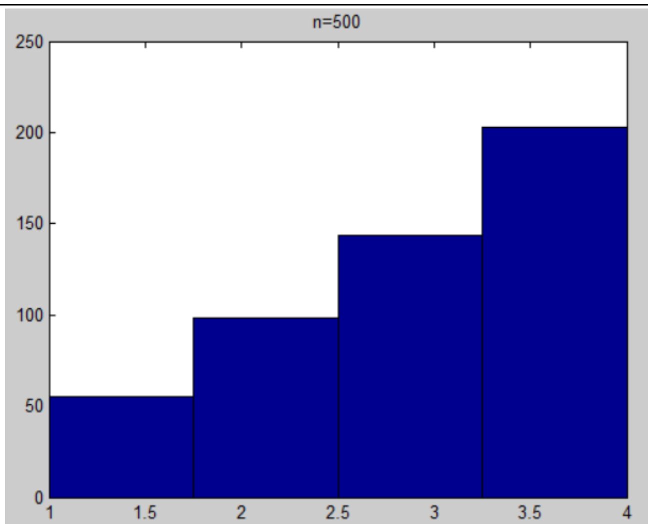
(1) CDF of w: [0   0.1   0.3   0.6   1]

(2) Randomly generates 10 numbers between 0 and 1.

(3) Put the numbers into the four gaps (as bins) of CDF(w).

(4) The ideal distribution would be 0.1, 0.2, 0.3, 0.4 respective.

N=100	N=200
	

N=300	N=400
	
N=500	
	

Problem2

Code:

```
%% problem 2 %%
load('cancer.mat');

trainX = X(2:10,184:end);
testX = X(2:10, 1:183);

trainLabel = label(184:end);
testLabel = label(1:183);

%% no boost
posTrain = trainX(:,trainLabel==1);
```

```
negTrain = trainX(:,trainLabel==-1);
muPosTrain = mean(posTrain,2);
muNegTrain = mean(negTrain,2);

trainMean = mean(trainX(:,:),2);
repTrainMean = repmat(trainMean,[1 500]);
sharedCov = (trainX-repTrainMean)*(trainX-repTrainMean)' ./ 500;
invcov = inv(sharedCov);
detcov = det(sharedCov);
correct = 0;

w0 = log(size(posTrain,2)/size(negTrain,2))-
(1/2)*((muPosTrain+muNegTrain)')*invcov*(muPosTrain-muNegTrain);
w = invcov * (muPosTrain-muNegTrain);

for te = 1:size(testX,2)
        sig = (testX(:,te)')*w+w0;
        if (sig*testLabel(te)>0)
                correct = correct+1;
        end
end

noboostAcc = correct/length(testLabel)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% boost

point3 = [101, 106, 121];
distri3 = zeros(1000,3);
avg = zeros(10,1);

for av = 1:10
    distri = repmat([1/500],[1 500]);
    epsilon = zeros(1000,1);
    alpha = zeros(1000,1);
    weakClassifier = zeros(500,1);
    w0 = zeros(1000,1);
    w = zeros(1000,9);
```

```
    %% boost
    for t = 1:1000
        sampleIndex = sampleDiscrete(500,distri);
        sampleSet = trainX(:, sampleIndex);

        samplePosIdx = sampleIndex(trainLabel(sampleIndex)==1);
        sampleNegIdx = sampleIndex(trainLabel(sampleIndex)==-1);

        samplePos = trainX(:,samplePosIdx);
        muSamplePos = mean(samplePos,2);
        sampleNeg = trainX(:,sampleNegIdx);
        muSampleNeg = mean(sampleNeg,2);

        w0(t) = log((size(samplePos,2))/(size(sampleNeg,2)))-(1/2) *
((muSamplePos+muSampleNeg)') * invcov * (muSamplePos-muSampleNeg); %1*1
        w(t,:) = invcov * (muSamplePos-muSampleNeg); %1*9

        weakClassifier = sign((w(t,:)*trainX)'+w0(t)); %(1*9)*(9*500)
        score = weakClassifier.*trainLabel';
        epsilon(t) = sum(distri(score<0));
        alpha(t) = (1/2)*log((1-epsilon(t))/(epsilon(t)));

        distri = distri.*exp(-1*alpha(t)*(score'));
        distri = distri/sum(distri);
        distri3(t,:) = distri(point3);
    end

    %boostAcc
    boostAcc = 0;
    boostCorrect = 0;
    for te = 1:size(testX,2)
        xte = testX(:,te);
        WX = w*xte+w0;
        alphaWX = sign(sum(alpha .* sign(WX)));
        if (alphaWX*testLabel(te)>0)
             boostCorrect = boostCorrect + 1;
        end
    end
```

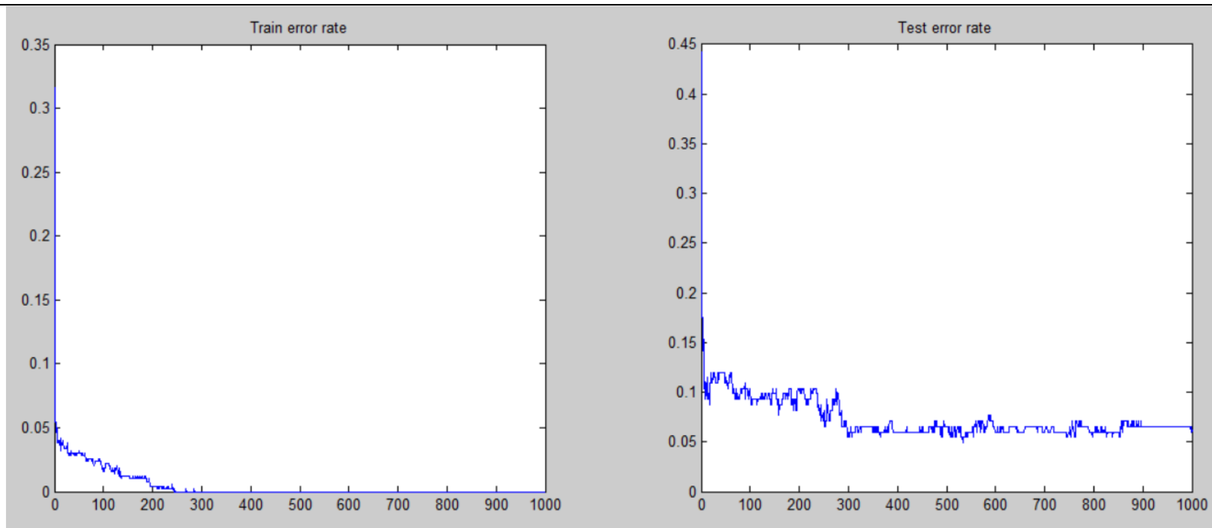
```
    boostAcc = boostCorrect / 183;
    avg(av) = boostAcc;
end
mean(avg)

trainErr = zeros(1000,1);
testErr = zeros(1000,1);
for ww = 1:1000
    pluginTrain = sign(w(1:ww,:)*trainX+repmat(w0(1:ww),[1 500])); % 1000*500
    alphaPluginTr = repmat(alpha(1:ww),[1 500]).*pluginTrain;
    resultTr = sign(sum(alphaPluginTr)).*trainLabel; % 1*500 X 1*500
    trainErr(ww) = length(resultTr(resultTr<0));

    pluginTest = sign(w(1:ww,:)*testX+repmat(w0(1:ww),[1 183])); % 1000*183
    alphaPluginTe = repmat(alpha(1:ww),[1 183]).*pluginTest;
    resultTe = sign(sum(alphaPluginTe)).*testLabel; % 1*183 X 1*183
    testErr(ww) = length(resultTe(resultTe<0));
end

trainErr = trainErr/500; testErr = testErr/183;

subplot(2,2,1), plot(alpha), title(['alpha']);
subplot(2,2,2), plot(epsilon), title(['epsilon']);
subplot(2,2,3), plot(trainErr), title(['Train error rate']);
subplot(2,2,4), plot(testErr), title(['Test error rate']);
%ssize = 1:1000;
%semilogx(ssize, trainErr);
%semilogx(ssize, testErr);

%subplot(1,3,1), plot(distri3(:,1)), title(['101 distribution']);
%subplot(1,3,2), plot(distri3(:,2)), title(['106 distribution']);
%subplot(1,3,3), plot(distri3(:,3)), title(['121 distribution']);
```

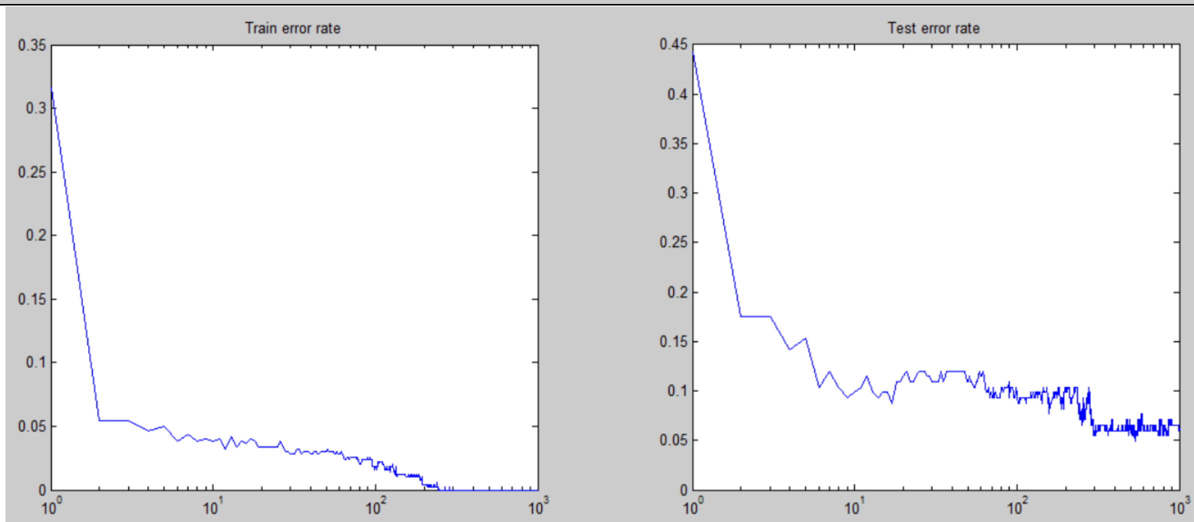
1. Boosted Accuracy (average of 10 times) : 92.13%

0.9235	0.9508	0.9290	0.9235	0.8962	0.8907	0.9235	0.9235	0.9126	0.9399

2. Training and Test error on iteration t:

Iteration times from 1-1000 on x and log-x axis respective.

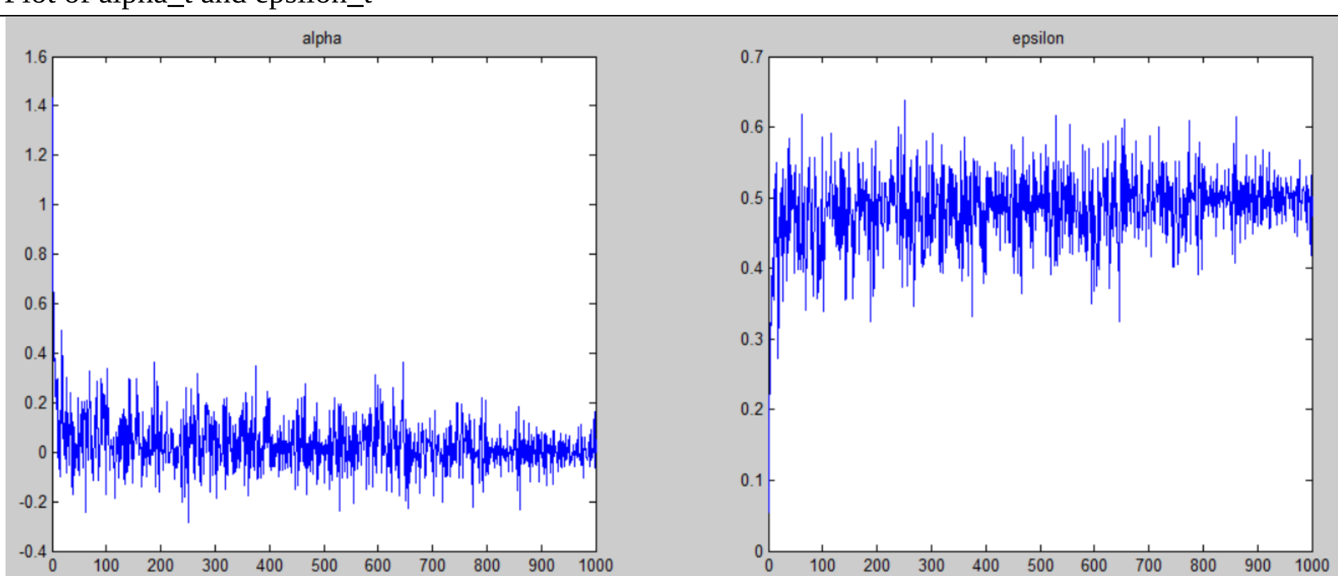




3. Accuracy without boosting (code is included above): 84.15%

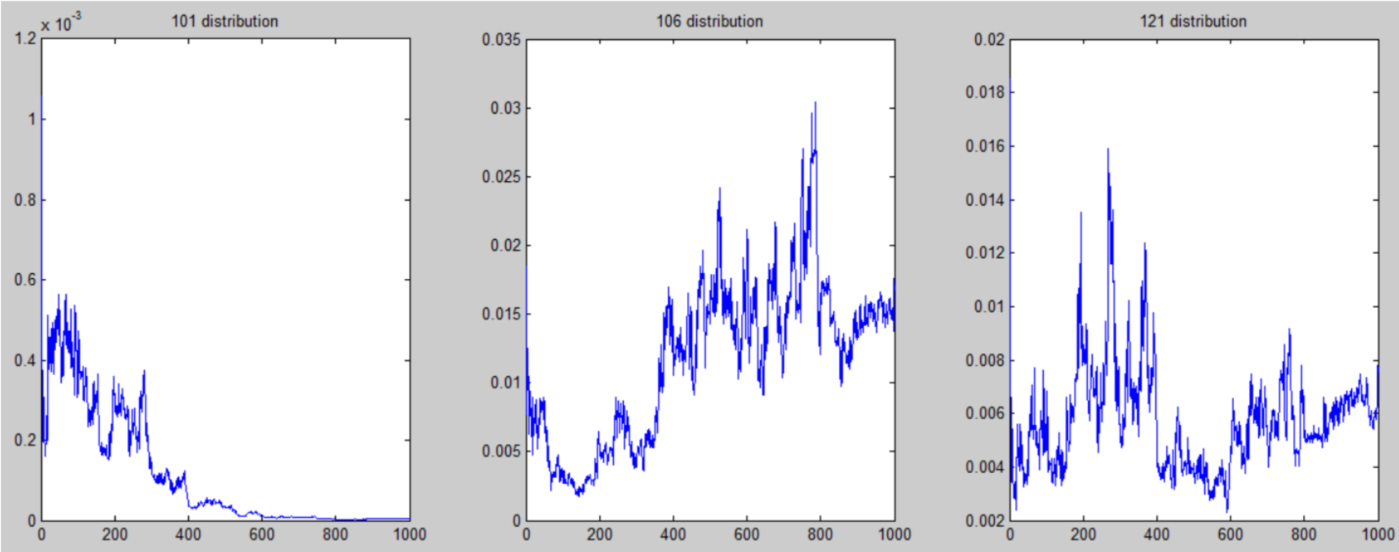
(This is always the same because of same training and test data without any randomness.)

4. Plot of alpha_t and epsilon_t



5. Distribution variation of three points randomly selected in training data set:

Data 101, 106, 121 in training set: (284, 289, 304 of 683)



Problem3

Code:

```
%% hw3pr3 %%
load('cancer.mat');

trainX = X(:,184:end);
testX = X(:, 1:183);

trainLabel = label(184:end);
testLabel = label(1:183);

%% no boost

avg = zeros(10,1);
for av = 1:10
    w = zeros(10,501);
    rp = randperm(500);
    newTrain = trainX(:,rp);
    newLabel = trainLabel(:,rp);

    for n = 2:501
        sigma = 1/(1+ exp(-1*newLabel(n-1) * (newTrain(:,n-1)') * (w(:,n-1))) );
        w(:,n) = w(:,n-1) + 0.1 * (1-sigma) * newLabel(n-1) * newTrain(:,n-1);
```

```
    end

    correct = 0;
    for te = 1:183
        es = exp((testX(:,te)')*w(:,501)); % 1*10 X 10*1 = 1*1
        fx = sign(es/(1+es)-0.5);
        if ((fx*testLabel(te)>0))
             correct = correct + 1;
        end
    end

    noboostAcc = correct/183;
    avg(av) = noboostAcc;
end
mean(avg)

%%%%%%%%%%%%%%%%
%% boost

point3 = [101, 106, 121];
distri3 = zeros(1000,3);
avg = zeros(10:1);

for av = 1:10
    boostW = zeros(10,1000); % for final boost
    recordW = zeros(10,501); % for online recording
    currentW = zeros(10,1);

    distri = repmat([1/500],[1 500]);
    epsilon = zeros(1000,1);
    alpha = zeros(1000,1);

    nextTrain = zeros(10,500);
    nextLabel = zeros(1,500);
    score = zeros(1,500);
    weakClassfier = zeros(500,1);

    for ite = 1:1000
        sampleIndex = sampleDiscrete(500,distri);
```

```
        nextTrain = trainX(:,sampleIndex);
        nextLabel = trainLabel(sampleIndex);

        for n = 2:501
             sigma = 1/(1+ exp(-1*nextLabel(n-1) * (nextTrain(:,n-1)') * (recordW(:,n-
1))) ); % 1*10 X 10*1
             recordW(:,n) = recordW(:,n-1) + 0.1 * (1-sigma) * nextLabel(n-1) *
nextTrain(:,n-1); % 10*1 = 10*1
        end
        boostW(:,ite) = recordW(:,501);
        currentW = recordW(:,501);

        exptrainXW = exp((trainX')*currentW); % 500*1
        fxtrain = exptrainXW./(1+exptrainXW);
        weakClassfier = sign(fxtrain-0.5);
        score = (weakClassfier').*trainLabel;
        epsilon(ite) = sum(distri(score<0));
        alpha(ite) = (1/2)*log((1-epsilon(ite))/(epsilon(ite)));

        distri = distri .* exp(-1*alpha(ite)*(score));
        distri = distri/sum(distri);
        distri3(ite,:) = distri(point3);
    end

     %boostAcc

    boostAcc = 0;
    boostCorrect = 0;
    for te = 1:size(testX,2)
        es = exp((testX(:,te)')*boostW); % 1*1000
        fx = sign((es./(1+es))-0.5);
        alphafx = sign(sum(fx*alpha));
        if (alphafx*testLabel(te)>0)
             boostCorrect = boostCorrect + 1;
        end
    end

    boostAcc = boostCorrect / 183;
    avg(av) = boostAcc;
```

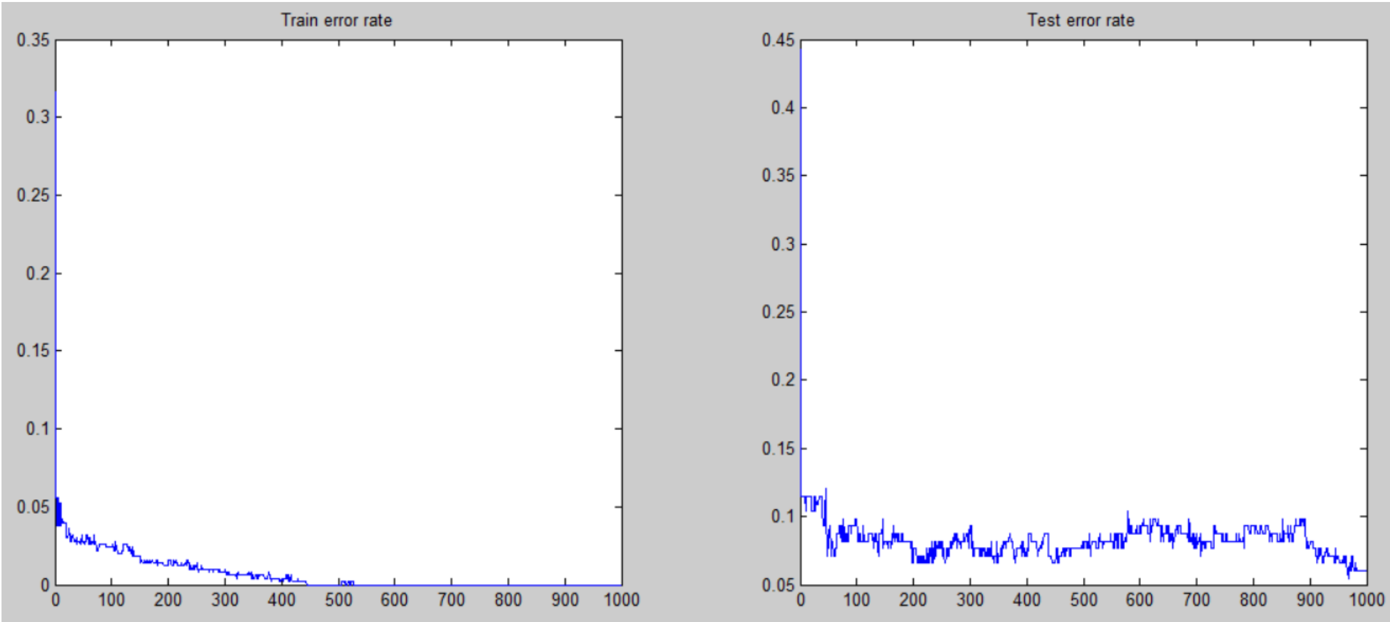
```
end
mean(avg)

trainErr = 0;
testErr = 0;
for ite = 1:1000
    prepareTrES = exp((boostW(:,1:ite)')*trainX); % ite * 500
    pluginTrain = sign((prepareTrES./(1+prepareTrES))-0.5);
    alphaPluginTr = repmat(alpha(1:ite),[1 500]).*pluginTrain;
    resultTr = sign(sum(alphaPluginTr)).*trainLabel; % 1*500 X 1*500
    trainErr(ite) = length(resultTr(resultTr<0));

    prepareTeES = exp((boostW(:,1:ite)')*testX); % ite * 183
    pluginTest = sign((prepareTeES./(1+prepareTeES))-0.5);
    alphaPluginTe = repmat(alpha(1:ite),[1 183]).*pluginTest;
    resultTe = sign(sum(alphaPluginTe)).*testLabel; % 1*183 X 1*183
    testErr(ite) = length(resultTe(resultTe<0));
end

trainErr = trainErr/500; testErr = testErr/183;

subplot(2,2,1), plot(alpha), title(['alpha']);
subplot(2,2,2), plot(epsilon), title(['epsilon']);
subplot(2,2,3), plot(trainErr), title(['Train error rate']);
subplot(2,2,4), plot(testErr), title(['Test error rate']);
%ssize = 1:1000;
%semilogx(ssize, trainErr);
%semilogx(ssize, testErr);

%subplot(1,3,1), plot(distri3(:,1)), title(['101 distribution']);
%subplot(1,3,2), plot(distri3(:,2)), title(['106 distribution']);
%subplot(1,3,3), plot(distri3(:,3)), title(['121 distribution']);
```

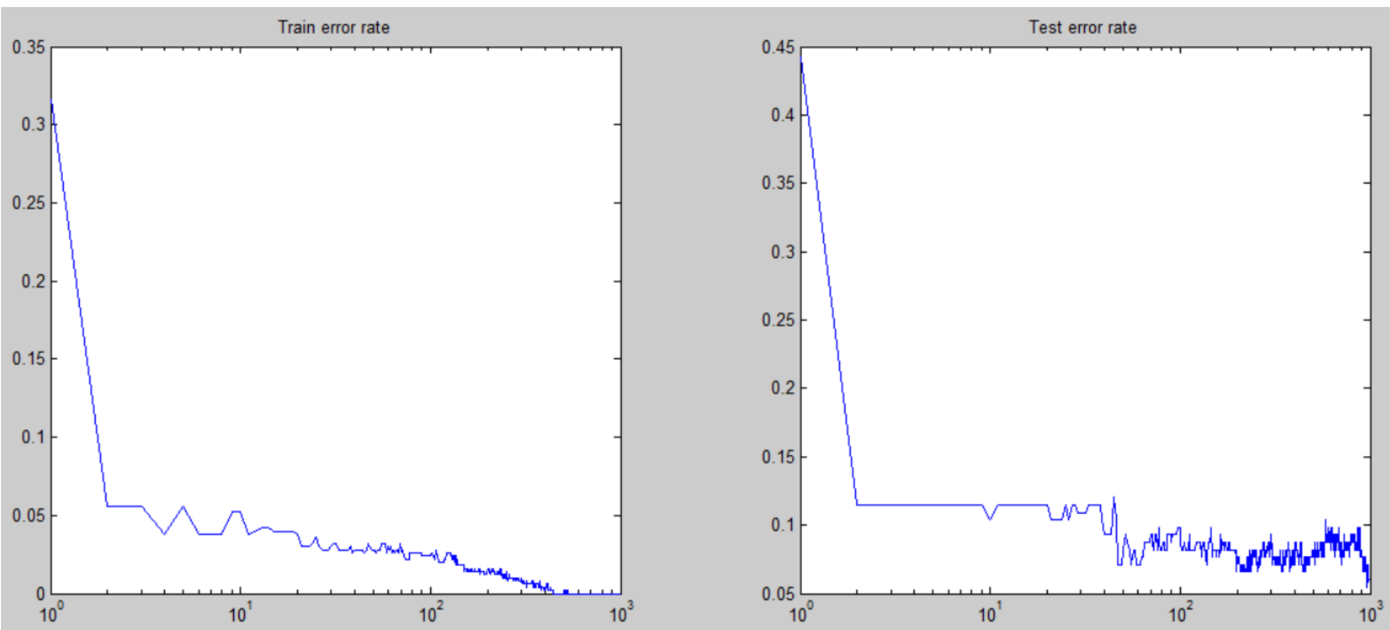
1. Boosted Accuracy (average of 10 times) : 93.55%

0.9508	0.9508	0.9454	0.9454	0.9126	0.9454	0.9071	0.9235	0.9344	0.9399

2. Training and Test error on iteration t:

Iteration times from 1-1000 on x and log-x axis respective.





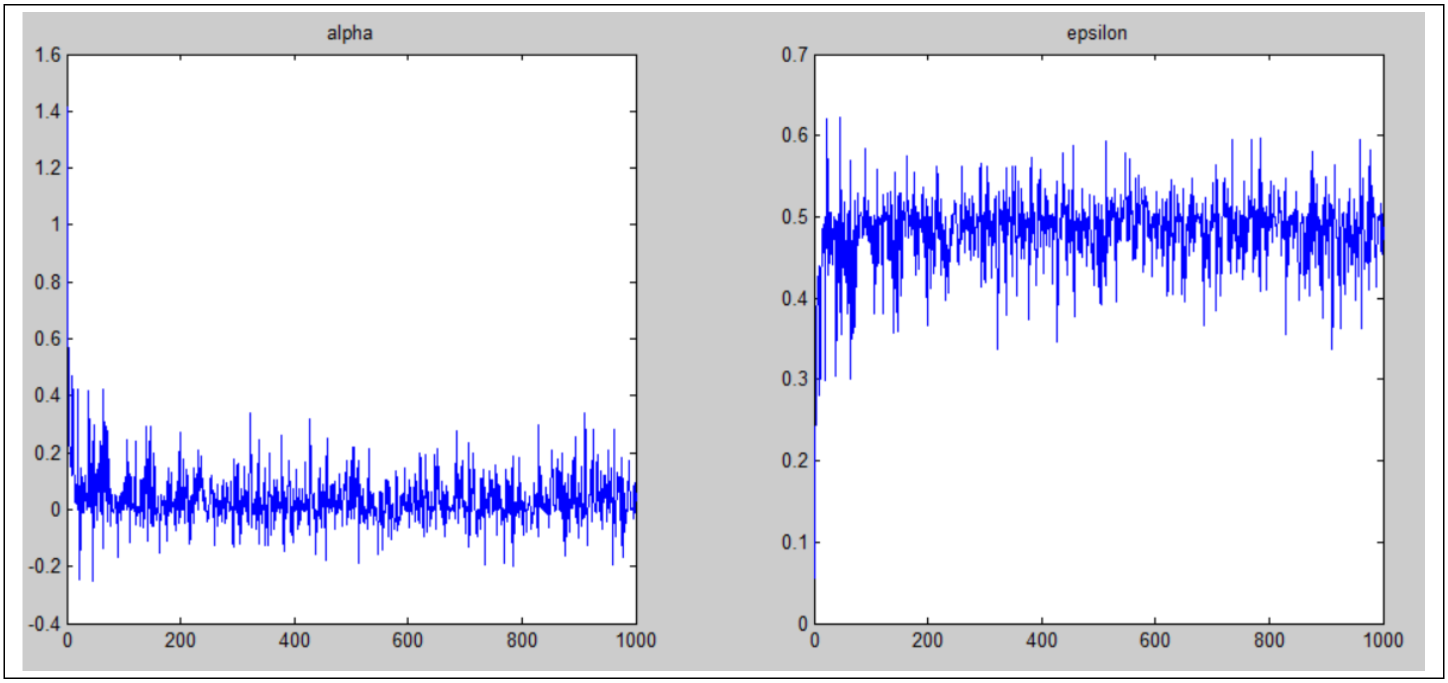
3. Accuracy without boosting (average of 10 times) (code is included above) : 84.86%

(This could vary due to the step coefficient η and the random order.)

0.9126	0.8634	0.9126	0.8795	0.8689	0.6831	0.8689	0.8525	0.7541	0.8907

Comparing with using boosting techniques, the classifiers are more unstable since it depend on one training set.

4. Plot of alpha_t and epsilon_t



5. Distribution variation of three points randomly selected in training data set:

Data 101, 106, 121 in training set: (284, 289, 304 of 683)