a. Prove that $e^{-x} \geq (1 - x)$

Here we assume $g(x) = e^{-x} + x - 1,\ g(0) = 0$

Calculating the derivative of g(x) with respect to x, gives $\frac{dg}{dx} = -e^{-x} + 1 \begin{cases} < 0 & when\ x < 0 \\ > 0 & when\ x > 0 \\ 0 & when\ x = 0 \end{cases}$

Therefore $g(x)$ is monotonically decreased $when\ x < 0$ and monotonically increased $when\ x > 0,\ x > 0$ is the local minimum. This is $g(x) \geq 0$ for $x \leftarrow R$. So $e^{-x} \geq 1 - x$

b. Implement a 'weak learner':

```
function [feature, theta, y]=weaklearning(X,label)
% The decision stump-a weak learner for classification.
% The classification error the trainingset has to be minimized
% Input: prescribed labeled datsets X with label
% Ouput: optimal feature, theta and sign
%%
% get the total number of data and number of features
[n,f]=size(X);
Error=[];
Y=[];
for i=1:f
    for j=1:n
        % sign=1 represents for >
        sign=1;
        theta=ones(n,1)*X(j,i);
        % check if the feature value is > theta
        label_pr1=X(:,i)-theta>0;
        label_pr0=X(:,i)-theta<0;
        error1=sum(abs(label_pr1-label));
        error0=sum(abs(label_pr0-label));
        if error1>error0
            sign=0;
            error=error0;
        else
            error=error1;
```

```
            end
            Error(j,i)=error;
            Y(j,i)=sign;
        end
    end
% find where the minimum Error for determining optimal parameters
[emin,I] = min(Error(:));
[I_row, I_col] = ind2sub(size(Error),I);
theta=X(I_row, I_col);
y=Y(I_row, I_col);
feature= I_col;
end
```

c. Test the implementation on the dataset generated by gendats from Prtools.

The weaklearning algorithm is implemented on the datasets generated by genedats from Prtools, which consist of two Gaussian distributions as $\mu1 = [0, 0]^T$ and $\mu2 = [2, 0]^T$. The scatterplot of the two datasets with label 0 and 1 can be found in Figure 1.

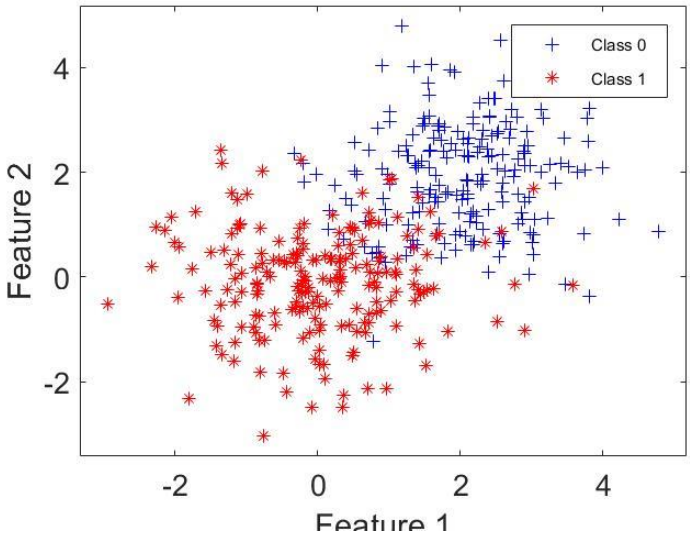

Figure 1. Scatterplot of the two datasets

The optimal parameters obtained by the decision stump can be determined as:

$$feature = 1, \quad \theta = 1.1343, \quad \text{sign=0(< for class 1)}$$

If I rescale the feature 1 by a factor, e.g. 10, the optimal feature and sign remain the same, while the theta is 10 times larger.

d. Test the implementation on dataset optdigitsubset of last week

Firstly we train the weak learning algorithm on the first 50 objects for each class, and consider the rest objects as testsets, giving a test error as 0.0176. When we add another 50 random objects to the training sets, the error of the testsets is found to be 0.0164, which did not change much. The mean and standard deviation of the error are determined as 0.1271 and 0.0164, respectively. The performance of the first 50 objects per class is unexpectedly bad for me, but without too much variance.

e. Extend the implementation of the weak learner such that it accepts a weight per object

To implement a weighted weak learner, I multiplied a weight matrix with the sum of errors. A simple dataset is constructed for testing, including 5 objects for each with label 0 and 1, as shown in Figure 2. The weighted weak learner was initialized with a weight matrix consisted of ones, and it gives the following optimal parameters as,

$$feature = 1, \quad \theta = 0.7969, \quad \text{sign=0(< for class 1)}$$

The observation in Figure 2 demonstrates that one object (x=0.3288) from class 0 was misclassified. If we assigned more weights to this misclassified object (9 times larger), the optimal parameters were found as,

$$feature = 1, \quad \theta = 0.3288, \quad \text{sign=0(< for class 1)}$$

Apparently, the optimal theta shifted towards the misclassified point to correct this misclassification. Therefore it can be convinced that point with higher weights could have larger influence on the weak learner.

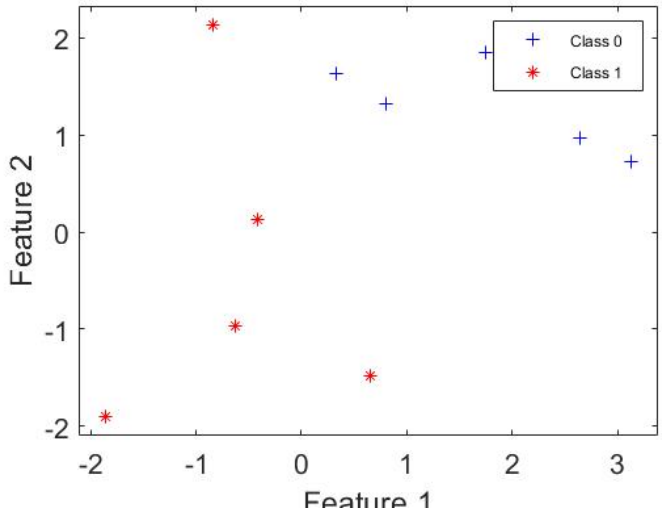

Figure 2. Scatterplot of the classes

f. Implement the algorithm that is described in Figure 2 of the paper: AdaBoost. Show the code.

```
function [W,e_ada,Hf]= adaboost(X,label,T)
% Input: labeled datasets X and label
%       Weak learning algorithm
%       Number of iterations T
% Output: Hypothesis, error and weights
%%
%Initilize w with equal weights
w_int=ones(size(label,1),1);
hf=zeros(size(label,1),1);
W=zeros(size(w_int,1),T);
for i=1:T
    %give distribution p
    p=w_int./sum(w_int);
    %Call weighted weak learner
    [emin,feature, theta, y]=weightedweak_ada(X,label,p);
    if y==0
        ht=X(:,feature)-theta<0;
    else
        ht=X(:,feature)-theta>0;
    end
    Beta=emin/(1-emin);
    w_int=p.*Beta.^(1-abs(ht-label));
    %providing hypothesis
    hf=hf+log(1/Beta)*ht-1/2*log(1/Beta);
    W(:,i)=w_int./sum(w_int);
end
    Hf=hf>=0;
    e_ada=sum(abs(Hf-label))/size(label,1);
end
```

g. Test your implementation on some dataset.

The implementation was firstly tested on the similar dataset gendats, as shown as Figure 3. Clearly as the datasets are highly overlapped with each other, the classification task is pretty difficult to perform. However, after training with Adaboost , the visualized weights show the expected trends where the weights of some

objects start increasing and eventually lead to non-uniform distribution of the weights. Besides, a more difficult dataset-banana dataset is tested, as shown in Figure 4. Similar results are observed that easily classified objects have less weights in the end while the hard objects have higher weights eventually.

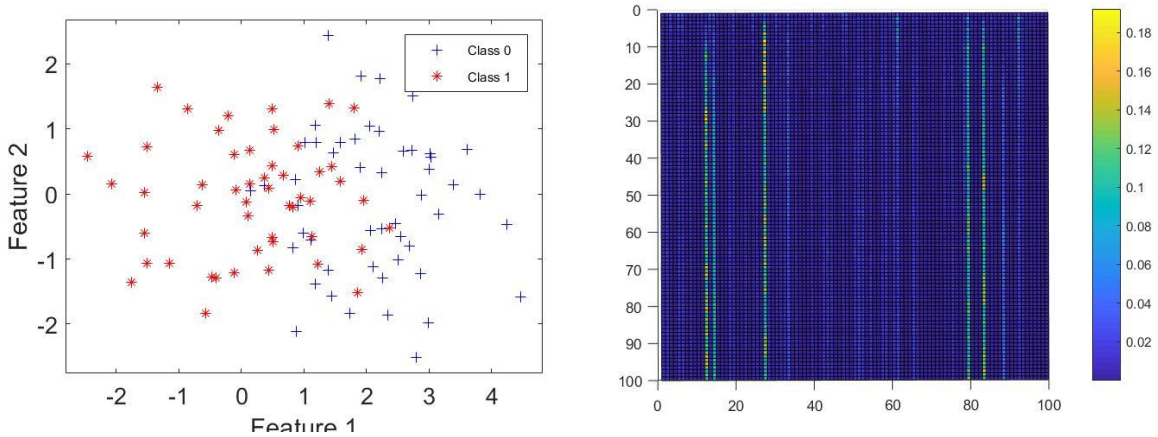

Figure 3 genedats datasets and visualizing weights for each object (X-axis) at each iteration(Y-axis)

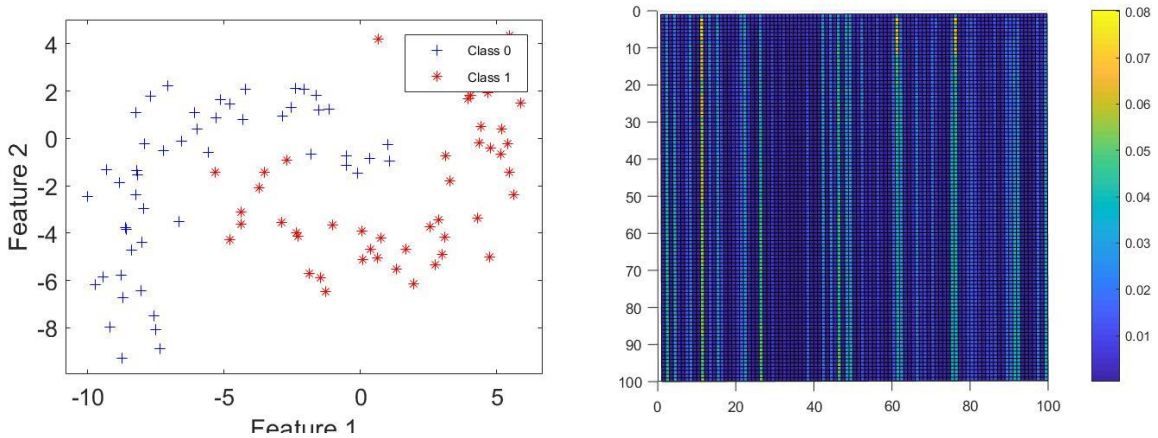

Figure 4 banana datasets and visualizing weights for each object (X-axis) at each iteration (Y-axis)

h. Test the implementation on dataset optdigitsubset of last week.

The implementation is tested on dataset optdigitsubset, giving the plot of error vs number of iterations (maximum 200) as shown in Figure 5. The minimum error is observed as 0.007805 at 17 iterations. The classification error dramatically drops with the number of iterations in the beginning and then starts to convergence with small oscillations. If I take the classifier with optimal T=17, objects 66 and 86 have much higher weights than the others as 0.467 and 0.25 respectively. The visualization of these images can be found in Figure 6. We can observe that these objects are difficult to distinguish since they are either distorted or abnormal. To counter the classification problem of these objects, the adaboost algorithm yields higher wrights to these objects with higher influences on the solution to enhance their importance.

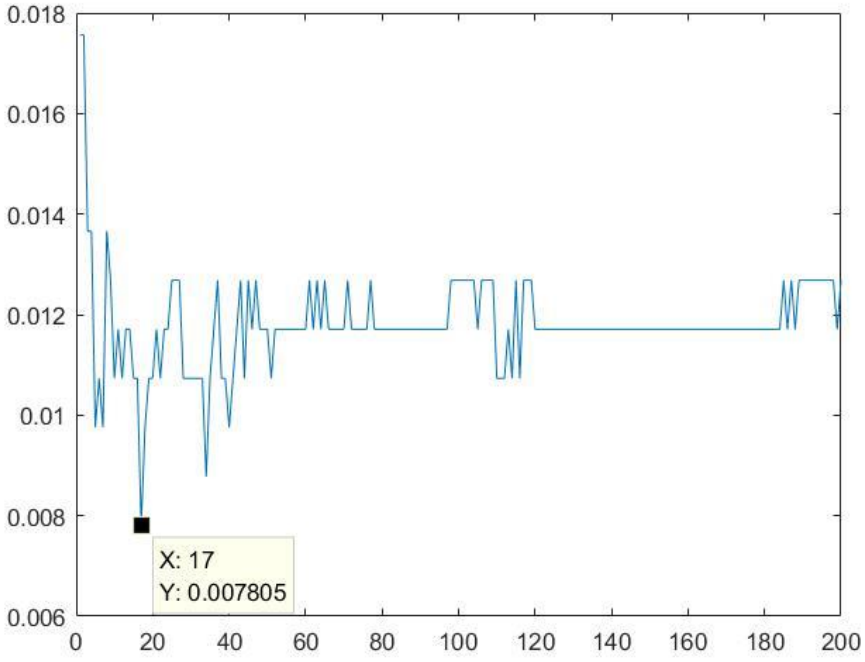

Figure 5 Error vs number of iterations.

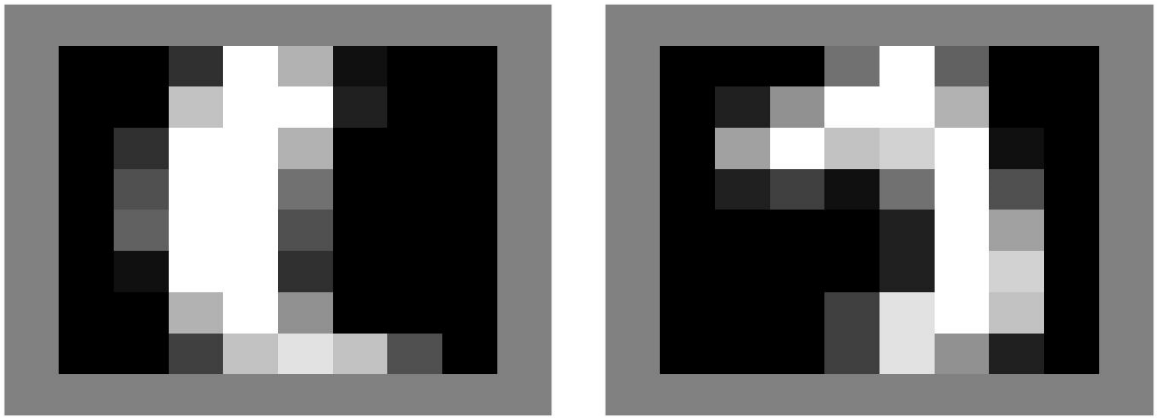

Figure 6 Visualization of object 66 and 86