IN4320 Machine Learning Computer Exercise
Computational Learning Theory: Boosting

Dilan Gecmen 4221168

June 1, 2018

a.

Suppose for the sake of contradiction that

$$e^{-x} < 1 - x$$

$$\iff$$

$$e^{-x} + x - 1 < 0$$

Let $f(x) = e^{-x} + x - 1$. The first derivative is $\frac{\partial f}{\partial x}(x) = -e^{-x} + 1$
Note that

$$\frac{\partial f}{\partial x}(x) \iff x = 0$$

The second derivative is $\frac{\partial^2 f}{\partial x^2}(x) = e^{-x}$. From the second derivative test it follows that f has a local minimum at $x = 0$ as

$$\frac{\partial^2 f}{\partial x^2}(0) = e^{-0} = 1 > 0$$

This results in a contradiction, since $x = 0$ is a local minimum, and in this case even a global minimum, as f is a composition of convex functions. It follows that the assumption that $e^{-x} + x - 1 < 0$ must be false and hence

$$e^{-x} + x - 1 \geq 0 \Rightarrow e^{-x} \geq 1 - x$$

proving the claim.

b.

```
function [ feature, theta, sign, Error ] = stump( X,y )
%Implementation of a 'weak learner': the decision stump.


[n,features] = size(X); %returns the size of matrix X in separate variables n and features
thetaexperiment = linspace(min(min(X)),max(max(X)),n*features*features);

Error = 100*n;
    for i = 1:length(thetaexperiment)
        Est1 = X<thetaexperiment(i); %Return a logical array with 1 if X<thetaexperiment(i) and 0 otherwise
        Est2 = X>=thetaexperiment(i); %Return a logical array with 1 if X>=thetaexperiment(i) and 0 otherwise
```

```
           Err1 = sum(Est1~=y)/length(y);
           Err2 = sum(Est2~=y)/length(y);
          [MinError1, f1] = min(Err1);
          [MinError2, f2] = min(Err2);

          if MinError1 < Error || MinError2 < Error
                if MinError1<MinError2
                     Error=MinError1;
                     feature=f1;
                     theta = thetaexperiment(i);
                     sign = 0; % The sign is <
                else
                     Error=MinError2;
                     feature=f2;
                     theta = thetaexperiment(i);
                     sign = 1; % The sign is >=
                end
          end
     end
end
```

c.

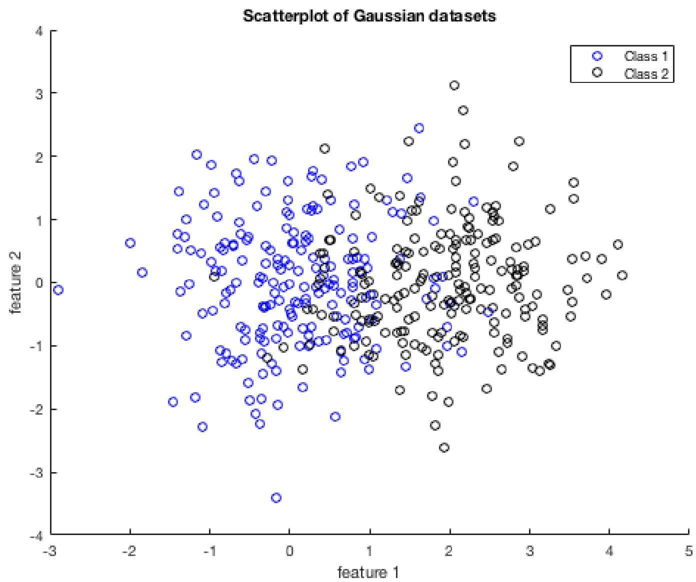

Figure 1: Scatter plot of two Gaussian datasets

The optimal parameters obtained by decision stump are:

- $f^* = 1$ the first feature is optimal
- $\theta^* = 0.9629$ is the optimal θ
- $y^* = 1$

The error is equal to 0.1485.
When we look at Figure 2 compared to Figure 1, we see that when we scale the first feature the distance between means is also scaled. This results in the θ also being scaled.
In Figure 3 we also scaled the mean. This affects the performance.
In Figure 4 we scaled the second feature. The results stay the as feature one is the optimal feature.

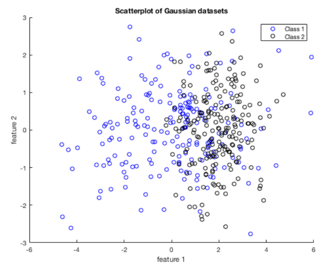

Figure 2: Scatter plot of two Gaussian datasets. Feature 1 scaled.

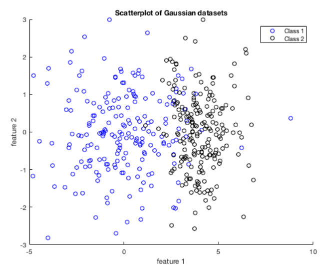

Figure 3: Scatter plot of two Gaussian datasets. Feature 1 and mean scaled.

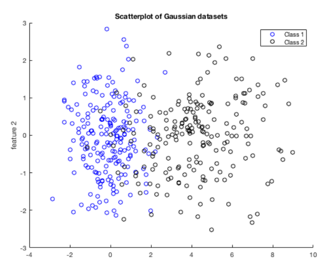

Figure 4: Scatter plot of two Gaussian datasets. Feature 2 scaled.

d.

The classification error on the test objects is:

- Test error = 0.0176

When we take random subsets of 50 for training we get the following mean and standard deviation of the classification error:

- Mean error = 0.0267
- Standard deviation error = 0.0098

The performance with training on the first 50 objects per class is good, as the results do not change a lot.

e.

```
function [ feature, theta, sign, Error ] = wstump(X,y,w)
%Implementation of a ‘weak learner?: the decision stump.
[n,features] = size(X); %returns the size of matrix X in
    separate variables n and features
thetaexperiment = linspace(min(min(X)),max(max(X)),n*
    features);

Error = 100*n;
     for i = flip(1:length(thetaexperiment))
           Est1 = X<thetaexperiment(i); %Return a logical
               array with 1 if X<thetaexperiment(i) and 0
               otherwise
           Est2 = X>=thetaexperiment(i); %Return a logical
               array with 1 if X>=thetaexperiment(i) and 0
               otherwise

           Err1 = sum((Est1~=y).*w)/length(y);
           Err2 = sum((Est2~=y).*w)/length(y);

          [MinError1, f1] = min(Err1);
          [MinError2, f2] = min(Err2);

          if MinError1 < Error || MinError2 < Error
                if MinError1<MinError2
                    Error=MinError1;
                    feature=f1;
                    theta = thetaexperiment(i);
                    sign = 0; % The sign is <
```

```
                else
                    Error=MinError2;
                    feature=f2;
                    theta = thetaexperiment(i);
                    sign = 1; % The sign is >=
                end
            end
        end
end
```

f.

```
% Import data set
A = importdata('optdigitsubset.txt');
q1 = zeros(554,1); q2 = ones(571,1);
y = [q1;q2];

% Test sets
Xtst = A;
Ytst = y;
Xtst(1:50,:) = [];
Xtst(555:604,:) = [];
Ytst(1:50,:) = [];
Ytst(555:604,:) = [];

% Training sets
Xtrn = [A(1:50,:) ; A(555:604,:)];
Ytrn = [y(1:50,:) ; y(555:604,:)];


Err = [];
Feat = [];
Theta = [];
Sgn = [];
ErrTst = [];

% Initialize for adaboost
B = []; p = [];
w = ones(length(Ytrn), 1);

k1 = zeros(length(Ytst),1);
k2 = zeros(length(Ytst),1);

n = 1000;


```

```
for i = 1:n

    p(:,i) = w(:,i)/sum(w(:,i));
    [ feature, theta, sign, Error ] = wstump( Xtrn, Ytrn,
        p(:,i) );

    B(i) = Error/(1-Error);

    if sign == 0
        r = Xtrn(:,feature) < theta;
    else
        r = Xtrn(:,feature) >= theta;
    end

    w(:,i+1) = w(:,i).*B(i).^(1-(abs(r-Ytrn)));

    Feat(i) = feature;
    Theta(i) = theta;
    Sgn(i) = sign;
    Err(i) = Error;

    if sign == 0
        tclf = Xtst(:,feature) < theta;
    else
        tclf = Xtst(:,feature) >= theta;
    end

    k1 = k1 + log(1/B(i));
    k2 = k2 + (log(1/B(i))).*(tclf);

    clf = k2 >= k1/2;
    ErrTst(i) = mean(clf~=Ytst); disp(ErrTst(i))

end
```

g.

We test the implementation for a fixed number of iterations $T = 1000$ on the complicated dataset gendatb, also known as the 'banana set' . We plotted the 10 objects i that obtain a large weight w_i^t. In Figure 5, these objects are depicted as the yellow circles.
Decision stump error is 0.019 and the error adaboost gives us is 0.0320. We see that Adaboost definitely performs better, and thus busted the performance of the weak learner
We see that the objects that obtain a large weight tend to lay between the two classes (depicted in blue and black). When the weight gets larger of the

objects depicted in yellow. The algorithm chooses these objects to classify them correctly to reduce the overall error. Falsely classifying an object with a large weight will enhance the error. This explains the yellow objects between the classes. See Figure 6.

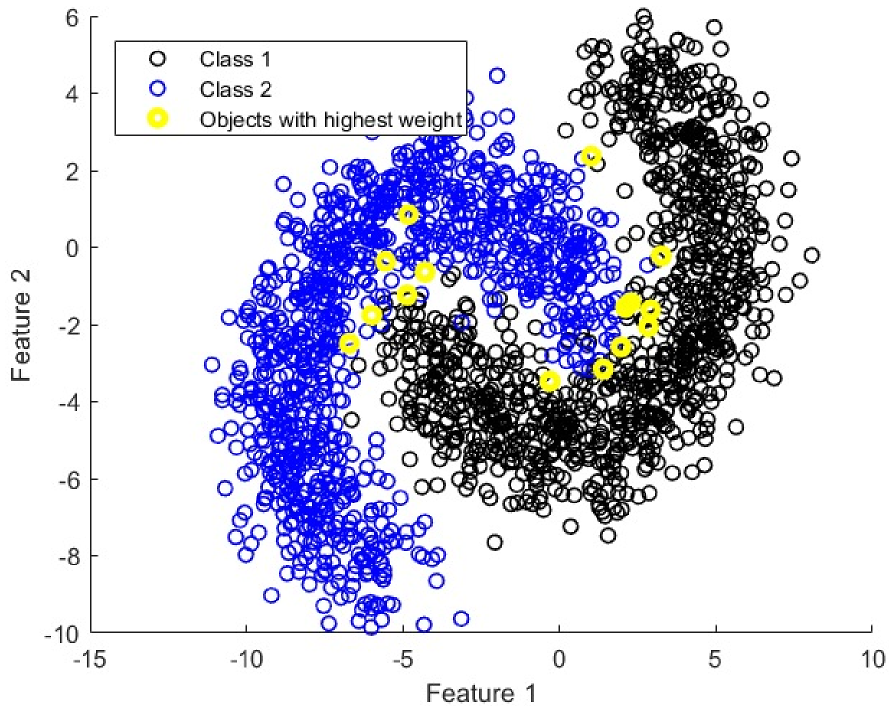

Figure 5: Scatter plots of the 'banana set' and the object with the largest weight.

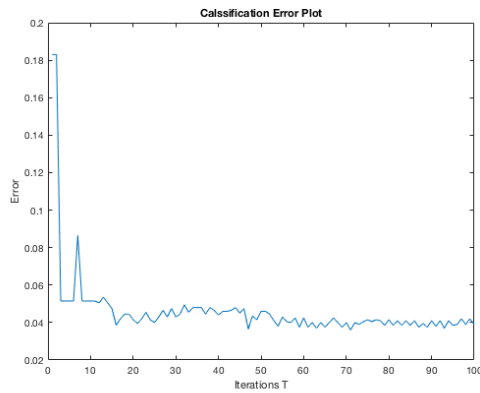

Figure 6: Plot classification error on the test objects.

h.

If we look at Figure 7 we see that Adaboost gives an error of 0.0078 at iteration $T = 17$.
In Figure 8 and 9 we can see images of a zero and a one. These are images of objects that contain the highest weight. These are one of the objects depicted in yellow in Figure 5 (I did not show the images of all 10 yellow objects only 2). From Figures 8 and 9 we can see that when the objects lay in between the borders of two classes. The distinction between a zero or a one can become a little vague.

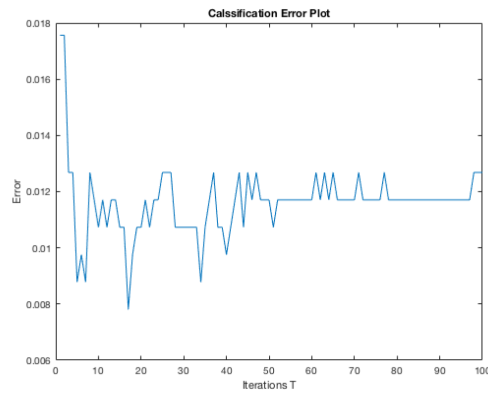

Figure 7: Adaboost plot classification error on the test objects.

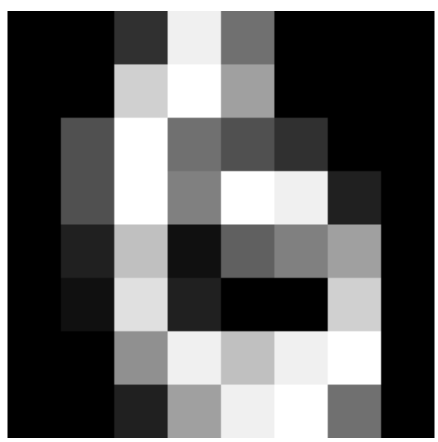

Figure 8: Plot zero highest weight.

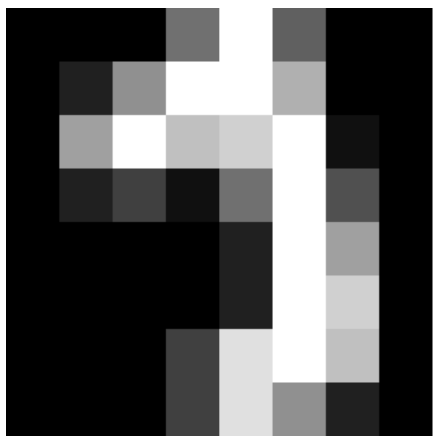

Figure 9: Plot one highest weight.