MACHINE LEARNING FROM DATA
Fall 2018

Report: Lab Session 6 – Decision Trees

Names: Santagiustina Francesco, Simonetto Adriano
Group:

Instructions

- Answer the questions
- Save the report and Matlab code in a folder, and upload the compressed folder (zip, rar).

Questions

Q1: Which are the default values of the tree depth controllers for growing the tree? (`MaxNumSplits, MinLeafSize, MinParentSize`).

MaxNumSplits corresponds to the maximum amount of branch nodes (nodes where a split between classes is made) that we can get for our tree. By default the parameter is set to the maximum possible amount, which is the number of observations minus 1. In our case this means the parameter is going to be 350.

MinLeafSize corresponds to the minimum amount of observations that each tree leaf has to have. The default value is 1.

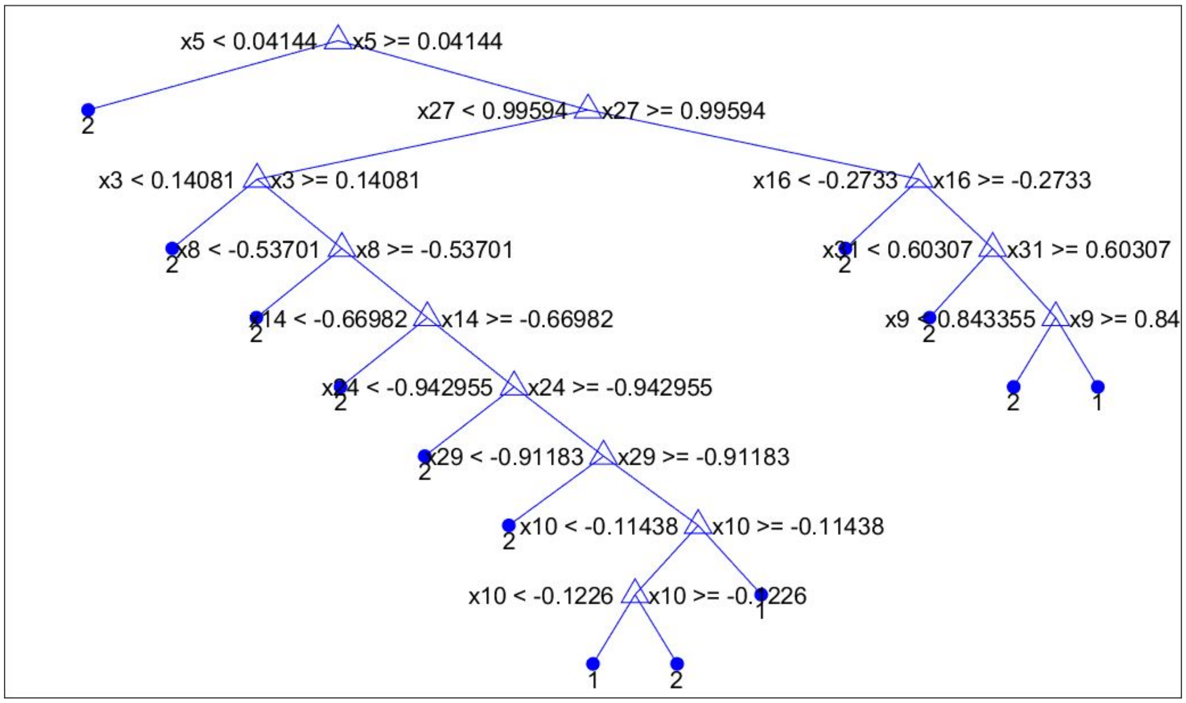
MinParentSize corresponds to the minimum amount of observations that a splitting node has to present. The default value is 10.

Q2: Run the script. Copy the training, validation and test errors, and the confusion matrices. From the *Classification Tree Viewer* copy the tree graph.

Training error: 0.0190
Validation error: 0.1
Test error: 0.0714

Confusion matrices:

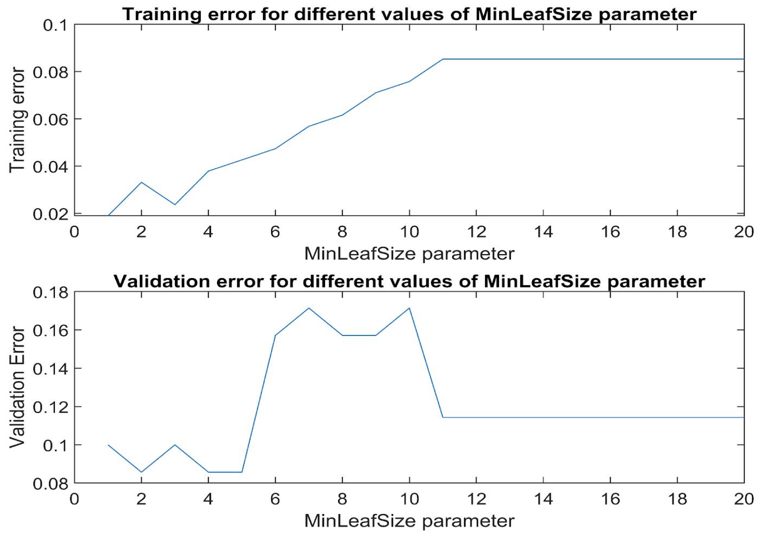
$$CM_{train} = \begin{bmatrix} 132 & 3 \\ 1 & 75 \end{bmatrix} \quad CM_{val} = \begin{bmatrix} 43 & 2 \\ 5 & 20 \end{bmatrix} \quad CM_{test} = \begin{bmatrix} 44 & 1 \\ 4 & 21 \end{bmatrix}$$



Q3: Analyze the questions at each node. Which are the most relevant features for the classification task?

The most relevant features are those appearing in the tree and thus being used for the classification process.
Looking at the picture above, they are: 5, 27, 3, 8, 14, 24, 29, 10, 16, 31, 9

Q4: Include error curves for training and validation subsets (for the different values of *MinLeafSize*), and the error and confusion matrices for the best classifier.



The best value for MinLeafSize turns out to be 2. Using said value we get a testing error of 0.0714 and the following confusion matrix:

$$CM_{test} = \begin{bmatrix} 44 & 1 \\ 4 & 21 \end{bmatrix}$$

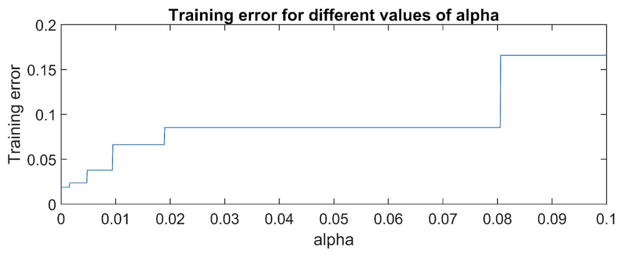
Q5: Include the code in the report.

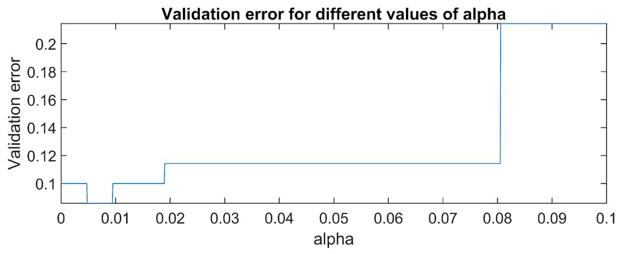
```
% Lab6 Decision trees
% VV ML_T2018
clear all;
close all;
clc;


load data_ionosphere % Contains X and XLabels variables

N_classes = 2;
N_samp = length(XLabels);

% training, validation and test indices
rng(1); % for reproducibility
P_train=0.6;
P_val=0.2;
P_test=1-P_train-P_val;
Index_train=[];
Index_val=[];
Index_test=[];

for i_class=1:N_classes
    index=find(XLabels==i_class);
    N_i_class=length(index);
    [I_train,I_val,I_test] = dividerand(N_i_class,P_train,P_val,P_test);
    Index_train=[Index_train;index(I_train)];
    Index_val=[Index_val;index(I_val)];
    Index_test=[Index_test;index(I_test)];
end
% Mixing of vectors not to have all belonging to a class together
Permutation=randperm(length(Index_train));
Index_train=Index_train(Permutation);
Permutation=randperm(length(Index_val));
Index_val=Index_val(Permutation);
Permutation=randperm(length(Index_test));
Index_test=Index_test(Permutation);
clear Permutation i_class index N_i_class I_train I_val I_test

% generation of training, validation and test sets
X_train=X(Index_train,:);
Labels_train=XLabels(Index_train);
X_val=X(Index_val,:);
```

```
Labels_val=XLabels(Index_val);
X_test=X(Index_test,:);
Labels_test=XLabels(Index_test);


minleaf = 1:20;
v_Pe_train = zeros(1,length(minleaf));
v_Pe_val = zeros(1, length(minleaf));
best_minleaf = 0;
best_val = realmax;
for i = 1:length(minleaf)
    % Tree classifier design
    tree =
fitctree(X_train,Labels_train,'MinLeafSize',minleaf(i),'Prune','off');
    %view(tree,'mode','graph');
    %view(tree)


    % Measure Train error
    outputs = predict(tree,X_train);
    Tree_Pe_train=sum(Labels_train ~= outputs)/length(Labels_train);
    %fprintf('\n------- TREE CLASSIFIER ------------------\n')
    %fprintf(1,' error Tree train = %g   \n', Tree_Pe_train)
    %CM_Train=confusionmat(Labels_train,outputs)
    % Measure Val error
    outputs = predict(tree,X_val);
    Tree_Pe_val=sum(Labels_val ~= outputs)/length(Labels_val);
    %fprintf('\n------------------------\n')
    %fprintf(1,' error Tree val = %g   \n', Tree_Pe_val)
    %CM_Val=confusionmat(Labels_val,outputs)

    v_Pe_train(i) = Tree_Pe_train;
    v_Pe_val(i) = Tree_Pe_val;
    if Tree_Pe_val < best_val
       best_val = Tree_Pe_val;
       best_minleaf = i;
    end
end
figure
subplot(2,1,1)
plot(minleaf,v_Pe_train)
title('Training error for different values of MinLeafSize parameter')
xlabel('MinLeafSize parameter')
ylabel('Training error')

subplot(2,1,2)
plot(minleaf,v_Pe_val)
title('Validation error for different values of MinLeafSize parameter')
xlabel('MinLeafSize parameter')
ylabel('Validation Error')
```

```
% Measure Test error
tree =
fitctree(X_train,Labels_train,'MinLeafSize',best_minleaf,'Prune','off');
outputs = predict(tree,X_test);
Tree_Pe_test=sum(Labels_test ~= outputs)/length(Labels_test);
fprintf('\n------------------------\n')
fprintf(1,' error Tree test = %g   \n', Tree_Pe_test)
CM_Test=confusionmat(Labels_test,outputs)



%%%%%%%%%%%%%%%%%%%%%%%%%

% Pre-prune / prune

% min_leaf = 1:20;
% alpha = 0:0.0001:0.1;

% TO DO, FOR MIN LEAVES AND TOP-DOWN CRITERIA

% for ParameterValues =
%   Train a tree with the train BD and the train targets
%   Measure Train, Val and Test classification errors
%   Keep or save the tree classifier associated to the minimum val
%   error
% end for
% Plot train, val and test errors for each value of the parameter

% END TO DO



%%%%%%%%%%%%%%%%%%%%%%%%%

% Train an ensemble
```

Q6: Include error curves for training and validation subsets (for the different values of *alpha*), and the error and confusion matrices for the best classifier.





The probability of error on the test set is 0.0714. The corresponding confusion matrix is:

$$CM_{test} = \begin{bmatrix} 45 & 0 \\ 5 & 20 \end{bmatrix}$$

Q7: Include the code in the report.

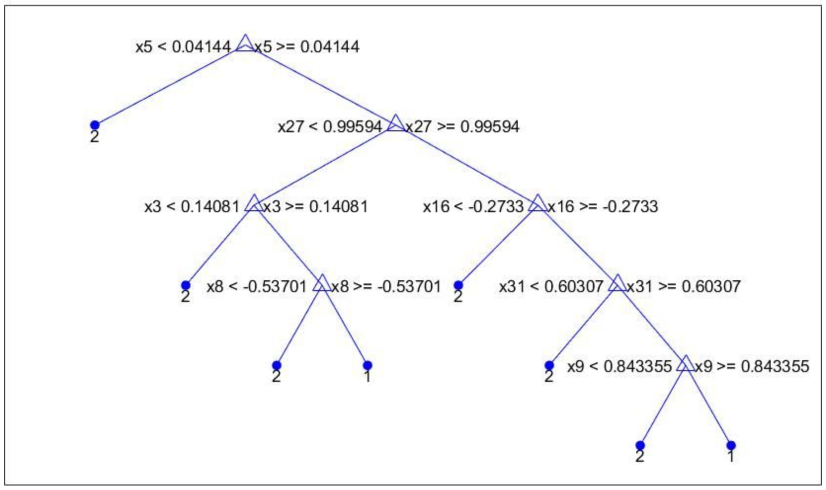
```
% Lab6 Decision trees
% VV ML_T2018
clear all;
close all;
clc;


load data_ionosphere % Contains X and XLabels variables

N_classes = 2;
N_samp = length(XLabels);
alpha = 0:0.0001:0.1;
% training, validation and test indices
rng(1); % for reproducibility
P_train=0.6;
P_val=0.2;
P_test=1-P_train-P_val;
Index_train=[];
Index_val=[];
Index_test=[];

for i_class=1:N_classes
    index=find(XLabels==i_class);
    N_i_class=length(index);
    [I_train,I_val,I_test] = dividerand(N_i_class,P_train,P_val,P_test);
    Index_train=[Index_train;index(I_train)];
    Index_val=[Index_val;index(I_val)];
    Index_test=[Index_test;index(I_test)];
end
% Mixing of vectors not to have all belonging to a class together
Permutation=randperm(length(Index_train));
Index_train=Index_train(Permutation);
Permutation=randperm(length(Index_val));
Index_val=Index_val(Permutation);
Permutation=randperm(length(Index_test));
Index_test=Index_test(Permutation);
clear Permutation i_class index N_i_class I_train I_val I_test

% generation of training, validation and test sets
X_train=X(Index_train,:);
Labels_train=XLabels(Index_train);
X_val=X(Index_val,:);
Labels_val=XLabels(Index_val);
X_test=X(Index_test,:);
Labels_test=XLabels(Index_test);




% Tree classifier design
tree = fitctree(X_train,Labels_train,'Prune','off');
%view(tree,'mode','graph');
%view(tree)
train_err = zeros(1,length(alpha));
val_err = zeros(1,length(alpha));
best_alpha = 0;
min_err = realmax;
for i = 1:length(alpha)
    tree1 = prune(tree,'alpha',alpha(i));
    % Measure Train error
    outputs = predict(tree1,X_train);
    Tree_Pe_train=sum(Labels_train ~= outputs)/length(Labels_train);
    %fprintf('\n------- TREE CLASSIFIER ------------------\n')
    %fprintf(1,' error Tree train = %g   \n', Tree_Pe_train)
    %CM_Train=confusionmat(Labels_train,outputs)
    % Measure Val error
    outputs = predict(tree1,X_val);
    Tree_Pe_val=sum(Labels_val ~= outputs)/length(Labels_val);
    %fprintf('\n-------------------------\n')
    %fprintf(1,' error Tree val = %g   \n', Tree_Pe_val)
    %CM_Val=confusionmat(Labels_val,outputs)
    train_err(i) = Tree_Pe_train;
    val_err(i) = Tree_Pe_val;
    if Tree_Pe_val < min_err
       min_err = Tree_Pe_val;
       best_alpha = alpha(i);
    end
end
```

```
% Measure Test error
tree = fitctree(X_train,Labels_train,'Prune','off');
tree1 = prune(tree,'Alpha',best_alpha);
outputs = predict(tree1,X_test);
Tree_Pe_test=sum(Labels_test ~= outputs)/length(Labels_test);
fprintf('\n------------------------\n')
fprintf(1,' error Tree test = %g   \n', Tree_Pe_test)
CM_Test=confusionmat(Labels_test,outputs)

figure
subplot(2,1,1)
plot(alpha,train_err)
title('Training error for different values of alpha')
xlabel('alpha')
ylabel('Training error')
subplot(2,1,2)
plot(alpha,val_err)
title('Validation error for different values of alpha')
xlabel('alpha')
ylabel('Validation error')



%%%%%%%%%%%%%%%%%%%%%%%%%

% Pre-prune / prune

% min_leaf = 1:20;
% alpha = 0:0.0001:0.1;

% TO DO, FOR MIN LEAVES AND TOP-DOWN CRITERIA

% for ParameterValues =
%   Train a tree with the train BD and the train targets
%   Measure Train, Val and Test classification errors
%   Keep or save the tree classifier associated to the minimum val
%   error
% end for
% Plot train, val and test errors for each value of the parameter

% END TO DO
%%%%%%%%%%%%%%%%%%%%%%%%%

% Train an ensemble
```

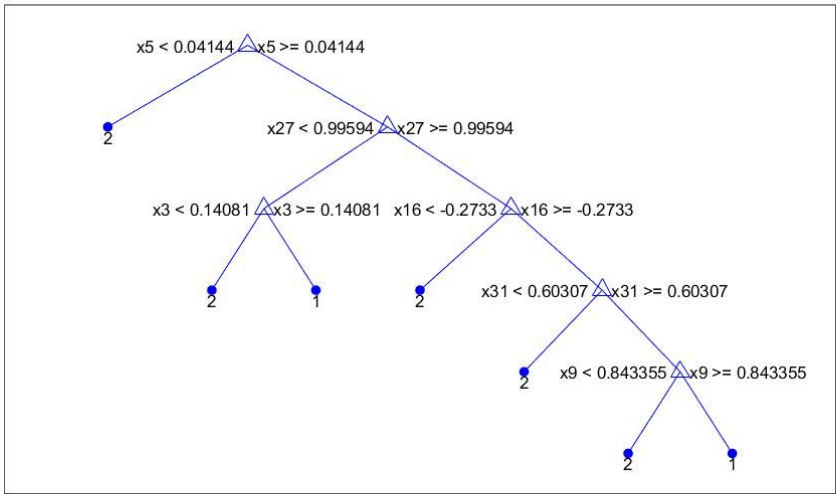
Q8: Compare results in Q4 and Q6. Which method is better?

We can see that in our case we obtain the same error probability (0.0714) for the test set, even if the misclassifications are not identical with both methods, as we can see from the confusion matrices. Generally speaking, top-down pruning, the second method, should give better results than pre-pruning, but at the cost of a higher computational complexity.

Q9: Analyze the trees obtained in Q4 and Q6. Which are the most relevant features? (Analyze questions in nodes)



From the hereinabove tree obtained by pre-pruning we can see that the most relevant are features number : 5, 27, 3, 8, 16, 31 and 9.



For the tree pruned with the top-down method we can see that we consider the same previous features : 5, 27, 3, 8, 16, 31 and 9 using also the same thresholds of the pre-pruned tree. Nevertheless one additional feature is taken into account : feature number 6, generating one more split in the tree.

Q10: How many trees are trained in the ensemble? Include errors and confusion matrices for training, validation and test subsets. Compare results with those obtained with a single tree (Q4, Q6).

By default the method `fitcensemble()` trains and boosts 100 classifications trees.
Using this model for predictions we get:

Training error: 0
Validation error: 0.0714
Test error: 0.0571

Confusion matrices:

$$CM_{train} = \begin{bmatrix} 135 & 0 \\ 0 & 76 \end{bmatrix} \quad CM_{val} = \begin{bmatrix} 45 & 0 \\ 5 & 20 \end{bmatrix} \quad CM_{test} = \begin{bmatrix} 45 & 0 \\ 4 & 21 \end{bmatrix}$$

We can see that performances have slightly increased with respect to both the top-down pruning and the pre-pruning methods, confirming the validity of this approach (anyway in our case it is questionable whether this small improvement is worth the increase in computational complexity). On the training set all the elements have been correctly classified and we could therefore fear some overfitting, but in practice the model generalizes well as we have pretty good performances also in the validation and test sets.