MACHINE LEARNING FROM DATA
Fall 2018

Lab Session 6 – Decision trees

1. Goal

The goal of this session is to

- Learn how to use decision trees to solve a classification problem
- Learn to prune a decision tree and build a simple ensemble of trees.

2. Instructions

Getting the material:

- Download and uncompress the file ML_Lab6_soft.zip

Handling your work:

- Answer the questions in the document Lab6_report_yourname.pdf
- Save the report and Matlab code in a folder, and upload the compressed folder (zip, rar).

3. Previous study

In this session we will use decision trees on the Ionosphere dataset, where the task is to classify radar returns from the ionosphere.

Read the slides corresponding to lecture 4.4: Decision trees.

4. Decision trees

4.1. Characteristics of the dataset

The dataset can be found here: https://archive.ics.uci.edu/ml/datasets/ionosphere

This is the information provided for the dataset:

```
This radar data was collected by a system in Goose Bay, Labrador. This
system consists of a phased array of 16 high-frequency antennas with a
total transmitted power on the order of 6.4 kilowatts. See the paper
for more details. The targets were free electrons in the ionosphere.
"Good" radar returns are those showing evidence of some type of
structure in the ionosphere. "Bad" returns are those that do not; their
signals pass through the ionosphere.

Received signals were processed using an autocorrelation function whose
arguments are the time of a pulse and the pulse number. There were 17
pulse numbers for the Goose Bay system. Instances in this databse are
described by 2 attributes per pulse number, corresponding to the
complex values returned by the function resulting from the complex
electromagnetic signal.
```

Attribute Information:

```
-- All 34 are continuous

-- The 35th attribute is either "good" or "bad" according to the
definition summarized above. This is a binary classification task
```

4.2. Classification using decision trees

Open the script `Lab6_trees.m` and identify the different sections in the code.
The function `fitctree` fits a binary decision tree for multiclass classification. We will first use it with the default parameters.

Answer the following questions:

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

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

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

Now we will try to reduce overfitting using two different strategies, pre-pruning, and top-down pruning (see slide #20).

Pre-pruning. Edit the script. Add the necessary code to select the best value for the `MinLeafSize` parameter: find the value that minimizes the error on the validation split. Try values of `MinLeafSize` between 1 and 20. When using the `fitctree` function set the parameter '*Prune*' to '*off*'. Compute the error curves for the training and validation splits. Compute the error and confusion matrices for the validation and test subsets for the best value of *MinLeafSize*.

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.

Q5: Include the code in the report.

Top-down pruning. Edit the script. Add the necessary code to perform a top-down pruning. In this case, the function to minimize is based on the sum of the errors and the number of leaves, and the parameter of interest is *alpha*, that weights the contribution of the two terms. First, fit the tree using `fitctree` with the parameter '*Prune*' set to '*off*'. Next, use the function '`prune`' to prune the tree (usage: `prune(tree,'alpha',`*alpha*`)`). Try values of *alpha* in the range `0:0.0001:0.1`. Find the best value of *alpha*. Compute the error curves for the training and validation splits. Compute the error and confusion matrices for the validation and test subsets for the best value of *alpha*.

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.

Q7: Include the code in the report.

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

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

Edit the script. Train an ensemble of decision trees. Use the function `fitcensemble` with the default parameters. Compute error and confusion matrices for training, validation and test subsets.

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).