Advanced computational methods - Problem set 5

Hrvoje Stojic

February 19, 2016

1. Develop your own function for classification trees. You can use mine from the handout as a starting point if you wish. I expect the following:

   - Function should have the following name: **cTree**.
   - It should be stored in a file **cTree.R**.
   - Function should be able to deal with multiple classes - I will check this by testing it on multi-class data.
   - Function should be able to deal with arbitrary number of dimensions.
   - Function should have a parameter `minPoints` for minimum number of points in a region, below that number the region should not be split.
   - Function should have a parameter for cost function, `costFnc`, that is used for growing the tree. Options should be `ME`, `Gini` and `Entropy`, as laid out in handout for lecture 7.
   - You should have at least the following arguments to the function: `formula` (formula object for specifying the dependent variable and predictors), `data` (dataset that formula will operate on), `depth` (integer, max number of nodes in each tree), `minPoints` (as described above), `costFnc` (cost function used for growing the tree, `ME`, `Gini` or `Entropy`, `Entropy` by default). Please use exactly these names. Of course, you can add more arguments if you need to.
   - Verify that inputs are in correct format (you can use `assertthat` package if you want).
   - Output should be a **named** list with at least these two elements - **predLabels** (vector of predicted labels for the observations in the dataset), **prob** (vector of probabilities for chosen labels). Please use exactly these names.
   - Do your best to come up with a more elegant and efficient solution than what I used in the class. Hint: use recursion.

- Finally, use SPAM dataset from the `Box/datasets` folder - generate training and test data, train cTree on the training data and evaluate it on the test data, do it for several different tree depths. Do the same with classification tree function from one of the packages and compare it with yours. You should produce `cTree.pdf` that illustrates the evolution of training and test errors for your function and from the package. Put this code in a file named **cTree_spam.R**.

2. (Optional, for ambitious ones) Using the `cTree` function from Problem 1 as a basis (you might need to modify it with the sampling of features mechanism), develop your own random forest algorithm.

   - Main function should have the following name: **randomForest**.
   - It should be stored in a file **randomForest.R**
   - The function should return out-of-bag error if instructed.
   - The function should return variable importances if instructed.
   - You should have at least the following arguments to the function: `formula` (formula object for specifying the dependent variable and predictors), `data` (dataset that formula will operate on), `noTrees` (integer, number of trees), `noFeatures` (integer, number of features subsampled in each iteration), `depth` (integer, max number of nodes in each tree), `OOB` (logical, compute out-of-bag errors or not, FALSE by default), `importances` (logical, compute relative importances of variables or not, FALSE by default), `costFnc` (cost function used for growing the tree, `ME`, `Gini` or `Entropy`, `Entropy` by default)
   - Output should be a **named** list with at least these two elements - **predLabels** (vector of predicted labels for the observations in the dataset), **prob** (vector of probabilities for chosen labels). Please use exactly these names.
   - Use again the SPAM data but now compare your randomForest function (implement also bagging) to the function from the `randomForest` package. You should produce `RF.pdf` that illustrates the evolution of training and test errors for these functions. Put this code in a file named **RF_spam.R**

Important notes

- Deadline is **February 28, at 12h**.
- Recall that you have to keep your code under version control in a repository at Github, I will pull the content of the repository at the designated time. Place all the files for this problem set in a folder in this repo under the name **PS5**.

- Create a simple text file with the name `Readme.md` in the problem set folder, if you want to leave me any instructions or messages regarding the problem set.

- Please, pay attention to the names of the files I instruct you to use. Names facilitate examining the code greatly, there is 30 of you and it takes me far more time if everybody produces different set of files with different names.

- I will give extra points for extra nice solutions!