Random forests

Tutorial/Lab work: Week 7

Instructions: This is a coding exercise. Kindly stage+commit+push the `MATLAB` code to the "`week06`" directory on your INM431 Github repository. Grading: All attempts will get 1 point. **Please make sure that the document is pushed to Github by 01/12/2021**.

Background

Random Forests build a collection of Decision Trees using bagging to divide the data for training among the different trees. A random choice of attribute (dimension) is made at each decision point (node) of each tree. In this assignment, we will use Information for the choice of threshold value to split the training data at each node.

The number of trees in the forest, the maximum depth of the trees, and a minimum number of examples per leaf node are hyperparameters of the Random Forest model.

Random forests can be used either for classification using majority voting or for regression by averaging results. The outcomes of the different trees can be combined to provide a probability distribution or a probability density function e.g. a mixture of Gaussians.

In `MATLAB` you can use in-built Random Forest functions e.g. `treebagger` and `fitcensemble` or implement your own custom Random Forest. This last option should give you more control over the implementation. You can also use and adapt the `MATLAB` code that is being provided with this tutorial.

In this tutorial, you are provided with a `MATLAB` implementation of Random Forests which you're asked to inspect and change a few hyperparameter values (as a way of making sense of the code provided to you). You're also pointed to a number of examples in `MATLAB` which use in-built functions.

Compare and contrast the different implementations before deciding which parts to use in case your choice of coursework includes Random Forests.

Setup
Download and unpack the `RandomForestCodeImplementation.zip` file from `Moodle`. This file contains an implementation of Random Forests, i.e. a collection of decision trees for regression and classification.
To test the Classification Forest run: `mainTreeClfUsage.m`. (Remember to add subfolder lib to the path)
To test the Regression Forest run: `mainTreeUsage.m`.
Notice that `mainTreeUsage.m` calls a number of other `.m` files.

Part 1

1. Inspect the code to find where in which file the number of trees can be changed.
2. Change the number of trees in the forest and run it again; record the results.
3. Inspect the code to find where you can change the maximum depth of the trees.
4. Change the size (maximum depth) of the trees in the forest and run it again; record the results.
5. Consider implementing a grid search consisting of evaluating the learning performance of the Random Forests for a few options of the number of trees and maximum depth.

Challenge: For the regression forest, calculate the mean and variance of each leaf node. Plot a sum of Gaussians which can be a measure of the forest's final regression result.

Part 2

The examples in thelink below illustrate the creation of an ensemble of decision trees both for classification and regression by using in-built functions in `MATLAB`:

`https://uk.mathworks.com/help/stats/treebagger.html`

Inspect the code trying to identify the main difference between the implementation used in Part 1 and the implementation of the above examples, both in the case of regression and classification.

The example in the link below uses Random Forests for regression with Bayesian Optimization for hyperparameter tuning.

`https://uk.mathworks.com/help/stats/tune-random-forest-using-quantile-error-and-bayesian-optimization.html`

Inspect the code and consider whether Bayesian Optimization might or might not be useful to your coursework.

Finally, the example in the link below creates a ranking of attributes (predictors). This ranking, which is a form of feature relevance, can be used to select among many attributes those top attributes for training a Random Forest on a subset of all predictors. Accuracy results can be compared for different subsets of predictors e.g. how accuracy decreases in relation to any speed-up gains when predictors are removed. Consider whether such accuracy vs. speed-up evaluation might be useful in your coursework. Feature ranking can also be used to help increase explainability of Random Forests, which is a popular current theme in ML.

https://uk.mathworks.com/help/stats/select-predictors-for-random-forests.html