MSc Stat. for Smart Data
2018-2019
matthieu.marbac-lourdelle@ensai.fr
fabien.navarro@ensai.fr

Homework of Parallel Computing with R and Python

Instructions

In this project, we want to implement (in R or in Python), a procedure for selecting the variables in the logit model. The procedure is a stepwise search (with the backward or forward direction) which optimizes the prediction error estimated by cross-validation (leave-one-out). All your answers should be included in a pdf (which can be generated by Markdown or by jupyter notebook). Your report should be submitted on Moodle before January, 6th.

About the logit model

Let $(X_1^\top, Y_1), \dots, (X_n^\top, Y_n)$ be observed independent copies of the random vector

$$(X, Y) \text{ with } X \in \mathbb{R}^d \text{ and } Y \in \{0, 1\}.$$

The distribution of Y given $X = x$ is assumed to be a logit model, such that

$$\mathbb{P}(Y = 1 \mid X = x) = \frac{e^{x^\top \beta}}{1 + e^{x^\top \beta}} \text{ and } \mathbb{P}(Y = 0 \mid X = x) = 1 - \mathbb{P}(Y = 1 \mid X = x),$$

where $\beta \in \mathbb{R}^p$ is the vector of the model parameters.

For prediction model, the procedures of selection of variables reduce the variance of the estimators. When p is large, it is convenient to assume that only r coefficients of β are not zero (with $r << p$). This leads to select a subset of the covariates as relevant for the prediction of Y. We said that a variable is relevant to predict Y if its coefficient is not zero.

Tasks

1. Implement the function `basic.mle` which takes as input argument the subset of the relevant variables and the variable to predict. This function returns the maximum likelihood estimator. This estimator is given by a Newton-Raphson algorithm (or any other algorithm of optimization). Detail your algorithm in the report. Use only basic instructions (like loops) and do not use any optimization techniques.

2. Implement the function `basic.cv` which takes as input argument the sample and the set of the relevant variables. This function returns the estimator of the error of prediction obtained by cross-validation for any subset of covariates by using the MLE of the model. Use only basic instructions (like loops) and do not use any optimization techniques.

3. Implement the function `basic.modelcomparison` which takes as input argument the sample and a set of competing models. This function returns the best model (*i.e.,* the subset of the relevant variables) and its estimator of the prediction error. Use only basic instructions (like loops) and do not use any optimization techniques.

4. Implement the function `basic.modelselection` which takes as input argument the sample and the direction (backward or forward). This function returns the best model (*i.e.,* the subset of the relevant variables) and its estimator of the prediction error. Use only basic instructions (like loops) and do not use any optimization techniques.

5. Explain, in the report, which parts of your code should be improved by using code monitoring. Explain, in the report, the evolution of the computing time according the sample size and to the number of covariates.

6. Propose different (optimized) versions of the functions `basic.mle`, `basic.cv`, `basic.modelcomparison` and `basic.modelselection` (at least one without parallelization and one with parallelization). Explain, in the report, which parts have been optimized.

7. Illustrate the gain (in computing time) of the optimized versions of your functions, in the report.