UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

ISSN 0103-2569

SIMPLIFIED FUNCTION TO APPLY THE BAYESIAN HIERARCHICAL STATISTICAL TEST

PAULO HENRIQUE PISANI
ANDRE C. P. L. F. DE CARVALHO

Nº 413

RELATÓRIOS TÉCNICOS

São Carlos – SP
Dez./2016

Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Simplified function to apply the Bayesian hierarchical statistical test

Paulo Henrique Pisani
André C. P. L. F. de Carvalho

Nº 413

ICMC TECHNICAL REPORT

São Carlos, SP, Brazil
December/2016

Simplified function to apply the Bayesian hierarchical statistical test

Paulo Henrique Pisani[1]
André C. P. L. F. de Carvalho[1]

[1]Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP) - Campus de São Carlos
13560-970 São Carlos - SP, Brazil
e-mail: {phpisani, andre}@icmc.usp.br

December, 2016

Abstract

The application of a statistical test to assess the experimental results is an important step in experimental research. Nowadays, the application of null hypothesis significance tests is very common in Machine Learning studies. However, the application of these tests in the area has been disputed due to several reasons. In view of these problems, a Bayesian hierarchical test was proposed in a recent tutorial. Based on the tutorial, this technical report presents a simplified code to apply the Bayesian test.

Keywords: statistical test, R, machine learning

Contents

1 Introduction 1

2 Requirements 1

3 The function 2

A Source code of the simplified function 6

1 Introduction

An important step in experimental research is the application of a statistical test to assess the significance of the obtained experimental results. Currently, the application of null hypothesis significance tests is very common in Machine Learning, such as *Friedman* and *Wilcoxon signed rank* tests (Demsar, 2006).

Nevertheless, the use of these tests to check whether a given algorithm performs significantly better than another algorithm has been criticized in the Machine Learning community (Demsar, 2008; Drummond, 2006; Corani et al., 2016; Benavoli et al., 2016). It is argued that null hypothesis significance tests imply in several drawbacks. A key argument is that these statistical tests simply do not output the probabilities of the null hypothesis and the alternative hypothesis, given the observed data. Instead, those tests return the probability of obtaining the observed or a larger difference between the algorithms being evaluated *if the null hypothesis was true*. It must be observed that it is different from returning the probability that one algorithm is better than another algorithm, given the observed experimental results.

In view of this scenario, a statistical comparison using Bayesian Hierarchical modeling was proposed by (Corani et al., 2016; Benavoli et al., 2016). The Bayesian test reports, *given the observed empirical results*, three probabilities: named $p(left)$, $p(rope)$ and $p(right)$. In the context of this report, they can be understood as the posterior probabilities that a proposed algorithm performs, respectively, worse, equivalent or better than a baseline algorithm.

This report describes a simple example to use the Bayesian Hierarchical test. The code was based on the original source made available by the authors os the test at `https://github.com/BayesianTestsML/tutorial` and a further exchange of e-mails with them. The tutorial from the authors of this Bayesian Hierarchical test can be found at `http://ipg.idsia.ch/tutorials/2016/bayesian-tests-ml/`. This technical report is based on the latest version of the code provided by the authors of the test (commit `2c1ea90` from 16/January/2017).

2 Requirements

The code that implements the test was executed on *R* 3.2.2 (`https://www.r-project.org/`) using *R Tools* 3.3 on MS Windows 10 (`https://cran.r-project.org/bin/windows/Rtools/`). In MS Windows, after installing *R Tools*, the paths `<Rtool-directory>\bin` and `<Rtool-directory>\gcc-4.6.3\bin` should be added to the PATH environment variable.

To run this code, the following packages from *R* are required:

- rstudioapi
- matrixStats
- rstan
- matrixcalc
- metRology

In addition, the *STAN* file shown next should be present at a folder named `stan` in the *R* working directory .

- $hierarchical - t - test.stan$

This file is the same provided by the tutorial from (Corani et al., 2016; Benavoli et al., 2016). The original version can be found at: https://github.com/BayesianTestsML/tutorial/blob/2c1ea90b801592212aec59b116b6b91f0967fac4/hierarchical/stan/hierarchical-t-test.stan.

3 The function

Based on the code provided by the authors of the test (Corani et al., 2016; Benavoli et al., 2016) at https://github.com/BayesianTestsML/tutorial/blob/2c1ea90b801592212aec59b116b6b91f0967fac4/hierarchical/hierarchical_test.R, a simplified example explaining how to apply the Bayesian Hierarchical Statistical Test function was written. The current version of the test can only be applied if the experiments were carried out using k-fold cross-validation.

The test was designed to compare the performance of two algorithms at a time (a baseline and a proposal, for example). For comparison of multiple algorithms, several tests must be performed. According to the authors (Benavoli et al., 2016), the false alarms due to multiple comparisons are mitigated by the adoption of a hierarchical model and the *rope* (the interval in which both algorithms are considered to be equivalent). The issue of multiple comparisons in this Bayesian Test may be further studied in the future.

The source code of the main function is shown in Appendix A. A simple function calling this main function adopting default parameter values is shown next, where *x* is a matrix of results and *numFolds* is the value of k used in the k-fold cross-validation. In the *x* matrix, the value of each cell is the difference of the metric (e.g. accuracy) for each run for all folds (columns) on each dataset (rows).

Simplified function using default parameter values.

```
hierarchical.test.default <- function(x, numFolds) {
  return( hierarchical.test(x, sample_file=NULL,
                            rho=(1/numFolds), chains=10) )
}
```

An example on how to use the previous function is shown next. This example simulates two classification algorithms (*good* and *bad*) tested under 5-fold cross-validation (30 runs per fold). The example can take a while to run (more than one minute in an average computer). Most of the time is spent by the main statistical test function.

Example to use the simplified function.

```
source('hierarchical_test.R')

runExample <- function() {
  set.seed(123)

  runsPerFold = 30
  numFolds = 5

  # Random good classification algorithm on 3 datasets (lets assume it is
      measuring the accuracy,
  # but it could be other metric, although their values always would be in
      the [0;1] range.)
  # Note: it seems that the model is designed for metrics with values in
      the [0;1] range
  goodClassificationAlg_Dataset1 = runif(numFolds * runsPerFold, min=0.9,
      max=1.0) # (number of folds) * (number of runs per fold)
  goodClassificationAlg_Dataset2 = runif(numFolds * runsPerFold, min=0.9,
      max=1.0)
  goodClassificationAlg_Dataset3 = runif(numFolds * runsPerFold, min=0.9,
      max=1.0)

  # Random bad classification algorithm on 3 datasets (lets assume it is
      measuring the accuracy,
  # but it could be other metric, although their values always would be in
      the [0;1] range.)
  # Note: it seems that the model is designed for metrics with values in
      the [0;1] range
  badClassificationAlg_Dataset1 = runif(numFolds * runsPerFold, min=0.6,
      max=0.9)
  badClassificationAlg_Dataset2 = runif(numFolds * runsPerFold, min=0.6,
      max=0.9)
  badClassificationAlg_Dataset3 = runif(numFolds * runsPerFold, min=0.6,
      max=0.9)
# CONTINUES IN THE NEXT PAGE
```

```
  # Compute the differences in the obtained values for the metric (e.g.
      accuracy)
  # IMPORTANT: all results must be obtained from the SAME DATA (pairwise),
      so fix the seed when running the cross-validation.
  # For example, train and test data for each run must be the same for all
      algorithms (pairwise)
  diff_Dataset1 = goodClassificationAlg_Dataset1 -
      badClassificationAlg_Dataset1
  diff_Dataset2 = goodClassificationAlg_Dataset2 -
      badClassificationAlg_Dataset2
  diff_Dataset3 = goodClassificationAlg_Dataset3 -
      badClassificationAlg_Dataset3

  # matrix x has 150 columns and 3 rows
  # From the example provided by the author, it seems that the runs are
      organized in this way:
  # [fold1run1, fold2run1, fold3run1, ..., fold5run1, fold1run2, fold2run2,
       ..., fol5run30]
  x = rbind(
      diff_Dataset1,
      diff_Dataset2,
      diff_Dataset3
    )

  tStart <- proc.time()
  retStatTest = hierarchical.test.default(x, numFolds)
  tFinish <- proc.time()
  print(tFinish - tStart)

  # >> nextDelta is used for inference on a next (unseen) dataset #
  print(paste("[nextDelta] Probability that the accuracy of
      goodClassificationAlg < the accuracy of badClassificationAlg = ",
      retStatTest$nextDelta$left, "(given the observed data on x)"))
  print(paste("[nextDelta] Probability that the accuracy of
      goodClassificationAlg = the accuracy of badClassificationAlg = ",
      retStatTest$nextDelta$rope, "(given the observed data on x)"))
  print(paste("[nextDelta] Probability that the accuracy of
      goodClassificationAlg > the accuracy of badClassificationAlg = ",
      retStatTest$nextDelta$right, "(given the observed data on x)"))

  # If an error rate is tested, the interpretation of the probabilities
      changes (lower is better)

  return(retStatTest)
}

runExample()
```

In the final part of the code, the results are printed. The output of the test are three probabilities: $p(left)$, $p(rope)$ and $p(right)$. The reported probabilities are used for inference on a next (unseen) dataset. In this example, they can be understood as follows:

- $p(left)$: the posterior probability that the accuracy of the *good* algorithm is worse than that of the *bad* algorithm;
- $p(rope)$: the posterior probability that the accuracy of the *good* algorithm is equivalent to that of the *bad* algorithm;
- $p(right)$: the posterior probability that the accuracy of the *good* algorithm is better than that of the *bad* algorithm;

The example compares the accuracy of two algorithms. It must be observed, however, that if an error rate is evaluated, the interpretation of the probabilities changes, since a lower value is better for error rates.

After running the code, the following output is obtained. Note that the probability of the performance of the *good* classification algorithm being better than that of the *bad* classification algorithm is higher than 95%. However, depending on the context, different threshold values can be set for an automatic decision, such as 90% or 80%.

```
Results obtained after executing the test.
```

```
[1] "[nextDelta] Probability that the accuracy of goodClassificationAlg <
   the accuracy of badClassificationAlg =  0.0218 (given the observed data
   on x)"
[1] "[nextDelta] Probability that the accuracy of goodClassificationAlg =
   the accuracy of badClassificationAlg =  0 (given the observed data on x)
   "
[1] "[nextDelta] Probability that the accuracy of goodClassificationAlg >
   the accuracy of badClassificationAlg =  0.9782 (given the observed data
   on x)"
```

Acknowledgements

The authors would like to thank Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP - processes 2012/25032-0 and 2013/07375-0).

References

Benavoli, A., Corani, G., Demsar, J., and Zaffalon, M. (2016). Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. CoRR, abs/1606.04316. Cited in page 1, 2, and 6.

Corani, G., Benavoli, A., Demsar, J., Mangili, F., and Zaffalon, M. (2016). Statistical comparison of classifiers through bayesian hierarchical modelling. Technical report, IDSIA. Cited in page 1, 2, and 6.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research, pages 1–30. Cited in page 1.

Demsar, J. (2008). On the appropriateness of statistical tests in machine learning. In Workshop on Evaluation Methods for Machine Learning in conjunction with ICML, pages 1–4. Cited in page 1.

Drummond, C. (2006). Machine learning as an experimental science (revisited). In Proceedings of the AAAI 2006 Workshop on Evaluation Methods for Machine Learning, pages 1–5. Cited in page 1.

A Source code of the simplified function

The original function for the Hierarchical statistical test was developed by the authors of the tutorial (Corani et al., 2016; Benavoli et al., 2016). However, the example described in this technical report uses a modified version of the original function provided by the authors of the test available at https://github.com/BayesianTestsML/tutorial/blob/2c1ea90b801592212aec59b116b6b91f0967fac4/hierarchical/hierarchical_test.R. The major change in the modified version is a code to fix the random seeds of the R and the STAN. The modifications are highlighted next.

Function adapted from the authors of the Bayesian test.

```
hierarchical.test <- function(x,
                              sample_file=NULL,
                              rho,
                              samplingType="student",
                              alphaBeta = list('lowerAlpha' =0.5,
                                               'upperAlpha'= 5,
                                               'lowerBeta' = 0.05,
                                               'upperBeta' = .15),
                              rope_min=-0.01,
                              rope_max=0.01,
                              std_upper_bound=1000,
                              chains=10,
                              stanSeed=123)
{

  set.seed(stanSeed)
  library(rstan)

  rstan_options(auto_write = TRUE)
  options(mc.cores = parallel::detectCores())
  library(matrixcalc)
  library(matrixStats)
  library(rstan)
  #for sampling from non-standardized t distribution
  library(metRology)
  set.seed(stanSeed)

...

  sample_file <- paste('stanOut/',sample_file,sep='')
  Nsamples <- dim(x)[2]
  q <- dim(x)[1]
  sample_file <- paste(sample_file,".StanOut",sep='')

...

    stanfit <-  stan(file = 'stan/hierarchical-t-test.stan', data =
       dataList,sample_file=sample_file, chains=chains,
    seed=stanSeed)

...

}
```