Bayesian comparison of two cross-validated algorithms on multiple data sets.

Giorgio Corani and Alessio Benavoli
giorgio{alessio}@idsia.ch

Technical Report No. IDSIA-01-15
January 2015

IDSIA / USI-SUPSI
Dalle Molle Institute for Artificial Intelligence
Galleria 2, 6928 Manno, Switzerland

This package allows to compare two algorithms whose performance has been assessed via cross-validation on multiple data sets. It performs a Bayesian correlated t-test on each data set and then merges their results via a Poisson-binomial inference. It returns the posterior probability of one algorithm having a higher mean score than the other on the provided collection of data sets. It accounts for the uncertainty and the correlation which characterize the cross-validation samples generated on each data set. We provide a Matlab and an R implementation.

Matlab implementation

The package can be used for two main goals:

- to compare two algorithms chosen by the user and assessed by cross-validation on multiple data sets. Such functions are located in the directory `general-usage`.
- replicating the experiments reported by Corani and Benavoli, (Machine Learning 2015). Such functions are located in the directory `replicating-experiments`.

The sub-directories of the package should be included in Matlab path.

Comparing two algorithms assessed by cross-validation on multiple data sets.

This function allows to compare two algorithms chosen by the user and assessed via cross-validation on multiple data sets.

```
[post_prob_h1]=compare_two_algorithms(dsets_results,n_folds)
```

Inputs `dset_results` is a matrix containing the difference of performance between two competing algorithms on each fold of each data set. Assume m runs of k-folds cross-validation are performed on each data set. Then `dset_results` has lenght $(m \cdot k \cdot q)$, where q is the number of data sets.

The matrix has two columns:

- the dset id;
- the *difference* between the score of the first algorithm and of the second algorithm.

`n_folds` is the number of folds adopted for cross-validation (assumed equal for all data sets). It is used to estimate the correlation of the cross-validation results on the same data set as 1/n_folds.

Output Returns the posterior probability `post_prob_h1`, namely the mean difference to be larger than 0 on the provided collection of data sets, accounting for the uncertainty of the cross-validation results. This is hence the posterior probability of the first algorithm having a higher mean score than the second on the provided collection of data sets.

Replicating experiments with fixed difference of accuracy on each data set

This function allows replicating the experiments reported by Corani and Benavoli (2015) characterized by a *fixed difference of accuracy* on each data set. See the paper for more details.

```
[results]=multiple_dsets_fixed_delta(delta_acc,how_many_dsets,n_runs,n_folds,reps)
```

where

- `delta_acc` is the actual difference of accuracy among the two classifiers, assumed identical for each data set.
- `how_many_dsets` is how many data sets are used in each experiment (parameter q of the paper);
- `n_runs`, `n_folds` are the number of runs and folds of cross-validation;
- `reps` is the number of repetitions of the experiments (we used 5000)

It returns the data structure `results` which contains all the raw results. Additionally a text file is created showing the power of the Poisson and the signed-rank test.

Experiments with Cauchy-distributed difference of accuracy on each data set

This function allows replicating the experiments reported by Corani and Benavoli (2015) characterized by a *Cauchy-distributed difference of accuracy* on each data set. See the paper for more details.

```
[results]=multiple_dsets_cauchy(delta_acc,how_many_dsets,n_runs,reps)
```

The parameters have the same meaning as above, apart from `delta_acc` which now represents the median of the Cauchy from which we sample the actual difference of accuracy to be used on each data set. Also in this case the returned results variables contains raw results. The produced text files summarize the power of the two tests in the performed experiments.

Replicating experiments on UCI data sets

```
>cd replicating-experiments
>load uci_analysis.mat
```

The workspace contain the results obtained by assessing 5 different classifiers by 10 runs of 10-folds cross-validation on 54 data sets. The results are stored in three variables: `Classifier_id,Dataset_ID,Percent_correct`. Each variable is a vector of length $5 \cdot 54 \cdot 10 \cdot 10$=27,000. The following function performs all the pairwise comparisons between couple of classifiers:

```
[decision_poisson,decision_sign_rank,p_value_sign_rank,post_prob_h1_poisson]=
analyze_uci(class_id,dset_id,exp_type)
```

You can compare the classifiers on the data sets 1-27, 28-54 or on the full collection of data sets, specifying the string '`first_half`','`second_half`','`full`' as `exp_type` parameter.

Each returned parameter is a matrix containing the outcome of the multiple comparisons. Only the out-of-diagonal elements are meaningful. Moreover the matrix is symmetric. Thus we report only the out-of-diagonal elements of the upper half of the matrix. Remaining elements in the matrix are set to `NaN`. The element (i,j) of the matrix is the results of the comparison between classifier i and j. When comparing classifiers i and j we look first at their average accuracy. If classifier i has higher average accuracy than classifier j, the direction of the test is as follows: H_0: classifier i has smaller equal mean accuracy than classifier j H_1: classifier i has larger accuracy than classifier j. The opposite direction is used if instead classifier j has higher average accuracy than classifier i.

The various matrices contains the decision and the posterior probability computed by our Poisson test and the decision and the p-values of the signed-rank test.

R implementation

The R implementation of the algorithm allows to compare two algorithms whose performance has been assessed via cross-validation on multiple data sets. The package is contained into the file *bayesTtest_1.0.tar.gz*. You install it as follows:

```
install.packages("bayesTtest_1.0.tar.gz", repos=NULL, type="source")
library(bayesTtest) # to load the package
```

Input data format

It requires the scores of the two algorithms measured via cross-validation on multiple data sets. These scores are stored into a list of numeric vectors: one vector for each data set. The vector contains the performance of each run and test fold of the cross-validation. For instance if there are two data sets, and a cross-validation performed on ten folds for ten run, the list would contain two numeric vector, each of size 100. The scores of the second algorithm must be stored in an identical structure.
Here an example of a possible input data, for two algorithms (named A and B) and two data sets (named *dataset1* and *dataset2*) with a cross-validation of two runs with two folds (therefore vector of size four):

```
> algorithmA=list(cv_dataset1=c(1,3,4,2), cv_dataset2=c(4,7,9,6))
> algorithmA
$cv_dataset1
[1] 1 3 4 2

$cv_dataset2
```

```
[1] 4 7 9 6

> algorithmB=list(cv_dataset1=c(2,7,8,6), cv_dataset2=c(5,9,9,8))
> algorithmB
$cv_dataset1
[1] 2 7 8 6

$cv_dataset2
[1] 5 9 9 8
```

Running the test

Once the data are ready, one has just to call the following function:

```
> bayesTtest(A = algorithmA, B = algorithmB,nFolds = 2)
$postH1
[1] 0.01205

$H
[1] 0
```

The function takes as input one list for the first algorithm and one for the second, as explained in the previous section, and the number of folds used in the cross-validation: As output it returns the a list of two items: the first item is the posterior probability of the alternative hypothesis (*postH1*), the second item is the decision of the test (H: 0 if the alternative hypothesis is rejected, 1 if the alternative hypothesis is accepted).

Contacts

Giorgio Corani and Alessio Benavoli are with IDSIA (`www.idsia.ch`).

Their homepages are `www.idsia.ch/~giorgio` and `www.idsia.ch/~alessio`.

Their e-mails are `giorgio@idsia.ch` and `alessio@idsia.ch`.

We thank Nicola Vermes `vermes@idsia.ch` who implemented the R version of the test.