Notes on the GP-Transferability paper

Roberto Santana Luis Marti Mengjie Zhang

1 Data processing

1.1 Description of the different steps and programs for GP transfer paper

- Downsample the brain signals to time series of 350 points.
- Reduce time series to 250 points describing the signal information after the stimulus onset.
- Remove outlier samples from the data.
- For each time series, create 9-feature vector formed by overlapping segments of 50 points with overlaps of 25 points.
- For each segment, compute two features (mean and slope of the segment).
- For each subject, use minimum-redundancy and maximum-relevance feature selection method to select 120 of the 5508 features.
- Feature normalization by subtracting the mean and dividing by the standard deviation.

The results of the preprocessing steps are included in the folder `data`, this folder should be included within the main folder.

1.2 Computation of the transfer measures

Using the RuLSIF algorithm[1] proposed by [1] we have computed the transfer measures for each pair of subjects. There are two transference metrics: rPE_i divergence values and each pair of source and targets (s,t), and density ratio of the univariate distributions for each possible value of variable X_i in the data set D_s.

The data structures with the divergences and density ratios are included in folder `matlab_data`. To run the optimizers, these two folders should be included within the main folder.

[1]The implementation is available from `http://www.makotoyamada-ml.com/RuLSIF.html`

2 Evolution of the single-objective classifiers

Evolution of the single objective classifiers is done using as objective function the classification accuracy in the train set. The program that implements this optimizer is `./SingleKalimeroTransferMEGGPDeap.py`. It can be called as:

`./SingleKalimeroTransferMEGGPDeap.py seed nvar 0 subj subj psize ngen` where $nvar = 120$, $subj$ is the subject number, and *psize* and *ngen* respectively represent the population size and number of generations.

3 Evolution of the MOP classifiers

Evolution of the bi-objective classifiers is done using one of the five bi-objective algorithms described in the paper. The program that implements these optimizers is `./KalimeroTransferMEGGPDeap.py`. It can be called as:

`./KalimeroTransferMEGGPDeap.py seed nvar Alg subj targetsubj psize ngen 1`

where the input parameters are as in the previous examples, and the parameter (Alg) to any of bi-objective algorithms is indicated as:

0 : Alg1, (O1,O2) Normal accuracy and transferability

1 : Alg2, (O2,O3) Biased accuracy and transferability

2 : Alg3, (O1,O3) Normal accuracy and biased accuracy

5 : Alg4, (O1,O4) LR (logistic regression) and normal accuracy

8 : Alg5, (O3,O4) LR (logistic regression) and biased accuracy

4 Evaluation of the other classification approaches

Classical classifiers are evaluated using grid search and 3 possible types of weight transfer for Importance Weighting Cross Validation. The program that runs these classical classifiers requires the instalation of the `libtlda` Python library [2], which is a library of transfer learners and domain-adaptive classifiers. However, currently this library implements only four classifiers. We extended some of the procedures in the library to deal with other classifiers.

The program `RevisedVersion_MEG_Problem_OtherClassifiers_v1.py` executes all combinations of classifiers, weight methods, and pair-wise transfer experiments. It can be called as:

`./RevisedVersion_MEG_Problem_OtherClassifiers_v1.py seed 120 3 1 2 1000 100 2 3`

[2] `https://github.com/wmkouw/libTLDA`

References

[1] M. Yamada, T. Suzuki, T. Kanamori, H. Hachiya, and M. Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Neural computation*, 25(5):1324–1370, 2013.