Learning a Bayesian network classifier by jointly maximizing accuracy and information

Learning a Bayesian network classifier by jointly maximizing accuracy and information [1,2,3] is a **generalization of the RMCV algorithm** [4] (see RMCV readme file). Similar to the RMCV, this code requires Kevin Murphy's Bayes Net Toolbox (BNT) [5], that is freely available from https://github.com/bayesnet/bnt.

The difference between the RMCV code and this code is the score function. While RMCV uses accuracy, this code allows more flexibility for *imbalanced ordinal classification problem*. The score functions implemented are:

$$Accuracy\ (ACC) = \frac{\sum_x \sum_y C_{x,y}\ ,\forall x=y}{\sum_x \sum_y C_{x,y}}$$

$$Mutual\ information\ (MI) = \sum_x \sum_y p(x,y) \cdot log\left(\frac{p(x,y)}{p(x) \cdot p(y)}\right)$$

$$Confusion\ entropy\ (CEN) = \sum_{j=1}^{N+1} \frac{\sum_{k=1}^{N+1}\left(C_{j,k} + C_{k,j}\right)}{2 \cdot \sum_k \sum_l C_{k,l}} \cdot \sum_{k=1,k\neq j}^{N+1} \left(P_{j,k}^{j} \cdot log_{2N}(P_{j,k}^{j}) + P_{k,j}^{j} \cdot log_{2N}(P_{k,j}^{j})\right)$$

$$Mattew\ correlation\ coefficient\ (MCC) = \frac{COV(U,V)}{\sqrt{COV(X,X) \cdot COV(Y,Y)}}$$

$$Mean\ absolute\ error\ (MAE) = \sum_x \sum_y p(x,y) \cdot |x-y|$$

$$Information\ measure\ (IM) = -MI(X,Y) + ES(X,Y,Errs) = \sum_{y=1}^{N} \sum_{x=1}^{N} p(x,y) \cdot \left(-log\left(\frac{p(x,y)}{p(x) \cdot p(y)}\right) + \log(1 + |x-y|)\right)$$

$$Information\ measure\ with\ alpha\ (IM_{\propto}) = \sum_{y=1}^{N} \sum_{x=1}^{N} p(x,y) \cdot \left(-log\left(\frac{\propto p(x,y)}{p(x) \cdot p(y)}\right) + \log(\propto (1 + |x-y|))\right)$$

$$= IM - \log(\propto) \times ACC$$

1. The algorithm can be found in the *extended_rmcv.m* file. The header of the main function is:

```
function [results, mats] = extended_rmcv(mytrain_data, mytest_data, class,
num_folds_k, init_type, score_type, varargin)
```

The inputs:

- *mytrain_data* and *mytest_data* are the train and test matrices (in a cell format, where rows are variables and columns are samples).
- *class* [integer] is the index of the target variable.
- *num_folds_k* [integer] is the number of folds for cross validation.

- *init_type* [string] is the initialized BN. Can take the values: 'E'-empty, 'D'-discriminative BN, and 'NBC'-naïve BN.
- *score_type* [string] can take one of the following values: 'ACC', 'MI', 'CEN', 'MCC', 'MAE', 'IM', 'IM_{alpha}' – indicating which score function to use.
- *varargin* can contain up to two input variables: *alpha* (relevant to IM_{α}) and *Real_MB* (relevant for synthetic datasets where the true structure is known and thus an SHD can be calculated).

The outputs:

- *results* [integer array] are the performance measures over the test set.
- *mats* [matrix array] are the confusion matrices for the training and test sets.

2. **The functions: *calculate_ACC/MI/CEN/MCC/MAE/IM/IM$_{\alpha}$* (within *extended_rmcv.m*) take as input a confusion matrix and return the relevant score. These functions can be used apart from the rmcv to learn or evaluate any classifier (e.g., decision tree, random forest).**

3. Demo:

   Steps to run the demo:

   3.1. Unzip the *demo.zip* folder which contains a *demo.m* file and *DBs* folder (to '*C:\Matlab*')
   3.2. Type in the Matlab command line:

```
run('C:\Matlab\demo.m')
```

References

1. D. Halbersberg and B. Lerner, "Learning a Bayesian network classifier by jointly maximizing accuracy and information", 22nd European Conference on Artificial Intelligence (ECAI), 2016.
2. D. Halbersberg and B. Lerner, "Young driver fatal motorcycle accident analysis by jointly maximizing accuracy and information", Accident Analysis and Prevention 129, pp. 350-361, 2019.
3. D. Halbersberg and M. Wienreb and B. Lerner, "Joint maximization of accuracy and information for learning the structure of a Bayesian network classifier", Machine Learning 110, pp. 1-61, 2020.
4. R. Kelner and B. Lerner, "Learning Bayesian network classifiers by risk minimization", International Journal of Approximate Reasoning, 53(2), pp. 248-272, 2012.
5. K. Murphy, "The Bayes net toolbox for Matlab", Computer Science and Statistics 33, pp. 331-350, 2001.