Case-based reasoning. Results and observations.

Aleksandra Piktus, Iosu Mendizabal Borda

December 1, 2013

1 HOW TO EXECUTE THE CODE

Our submission is organized as follows. The main folder contains 5 subfolders: `data`, containing the `.arff` files, `lib`, containing external code, `references`, containing pdfs with sources we used, `ten_fold`, containing folders with datasets, each divided into 10 folds and `src`, containing matlab scrips with our implementation of the CBR classisfier. The main folder contains the `graph.jpg` - the visualization of CBR algorithms' performance and this file (`report.pdf`).

In order to perform the classification task, the user is asked to execute the `main.m` script, residing in the `src` folder. The script executes all possible combinations of CBR algorithm on three datasets (`primary-tumor`, `glass`, and `iris`) and evaluates them. Next, the best combination is used to perform CBR classification with feature selection.

Other scripts included in the src folder are:

- `weka_reader.m` - the data parser (deals both with numerical and nominal data);
- `normalizer.m` - implements data normalization;
- `cbrClassifier.m` - main classification framework, parses and normalizes the data and runs the CBR algorithm for each fold of a given dataset, and evaluates the accuracy of all fold;
- `cbrAlgorithm.m` - implements the CBR algorithm for a single fold;
- `cbrRetentionPhase.m` - implements retention policies;
- `cbrRetrievalPhase.m` - implements similarity measures, finds the train instances similar to a given test instance;
- `cbrReusePhase.m` - implement reuse policies;
- `cbrRevisionPhase.m` - evaluates a single fold;
- `RetentionPolicies.m` - retention policies enumeration;
- `ReusePolicies.m` - reuse policies enumeration;
- `SimilarityMeasures.m` - similarity measures enumeration.

2 LIBRARY COMPILATION

One of the feature selection methods we use comes from the library found on the following website: `http://www.cs.man.ac.uk/~gbrown/fstoolbox/`. The code of the library can be found in the `lib/FEAST` folder, and it requires precompilation. In order to compile functions from FSToolbox and MIToolbox in Linux environment:

- compile Linux C shared library - use the included makefiles: `lib/FEAST/MIToolbox/Makefile` and `lib/FEAST/FSToolbox/Makefile`.
- run `./lib/FEAST/MIToolbox/CompileMIToolbox.m`;
- run `./lib/FEAST/FSToolbox/CompileFEAST.m`.

The user is asked to adapt the above commands to his/her operating system.

3 DATA PREPROCESSING

We have implemented an `.arff` file reader based on weka methods for reading `.arff` files, in order to receive easily all the information of the datasets and to change them in case of the nominal datasets.

With the weka method we load the `.arff` file and extract a struct with all the information on the dataset. The reader allows us to read nominal, numerical and mixed datasets. In case of the nominal data, the method reads the attributes of the datasets and changes their names to numbers: starting from zero and adding one for each value.

For example: In the case of the `primary-tumo`r datasets, we have the first attribute which is the age, and three different nominal values `{'<30','30-59','>=60'}`. The reader parses the data to the folloing form: `{0, 1, 2}`. So in all the cases of the datasets that the age of a row is âĂŸ30-59âĂŹ the age would be changed for the number 1.

Also we implemented a data standardization to deal with missing data in the datasets. In case of nominal data sets we replace the missing data with the most frequent data in the class, and in the case of numerical data we replace the missing data with the mean of the class.

4 CASE-BASED REASONING (CBR)

We apply the cased-based reasoning technique to perform a classification task on the datasets from the UCI repository. The CBR classification algorithm includes 4 phases:

- **retrival**;
- **reuse**;
- **retention**;
- **revision**.

We consider multiple combinations of different implementations of retrieval, reuse and retention phases (the details will be discussed in the following subsections). We use 10-fold cross-validation to measure the accuracy of each combination.

Once we have the accuracies of all the possible algorithms (all combinations of the 4 phases), we compare them with the statistical evaluation tools. The best CBR algorithm is then used to find out which of the two implemented feature selection algorithms is better.

4.1 Retrieval phase

This step is responsible for extracting K train instances (K rows of the train matrix), which are most similar to the current test instance (the row of the test matrix which is currently being classified).

To measure the similarity between the instances, we implement 2 basic measures:

- **Eucledean distance**;
- **cosine similarity**;

Initially we only experiment with the above measures. Once we have select the best CBR algorithm, we use its settings with the weighted similarity measures to compare the performance of the feature selection methods (details later).

We've decided to set the default K value to 5, as we observed that greater values don't improve the classification accuracy, but instead make the algorithm less time efficient.

4.2 Reuse phase

In the reuse phase we need to classify the current test instance based on the similar instances, extracted in the previous step. In other words, we need to choose the class, to which the test instance is going to be assigned.

The class inference may be performed with respect to one of the two following policies:

- **closest** - this policy selects the most similar instance (of all K similar instances, selected in the previous step), and assigns the current test instance to the class it belongs to;
- **most popular** - finds the class label, which is most numerous among the K similar instances, and assigns the test instance to it.

4.3 Retention phase

Once we have predicted the class of the current test instance, we have to decide what to do with this information. We implement three retention policies:

- **full** - adds all the classified test instances to the train set;
- **only correct** - adds only those test instances, which were correctly classified;
- **none** - doesn't retain any instances.

Thanks to the retention phase, our train set grows in train of the execution and possibly, the classification accuracy grows. There's however the risk of introducing wrongly classified instances with the full retention policy.

4.4 Revision phase

We don't implement the actual revision phase. Instead, once we have finished classifying all test instances, we calculate the accuracy of the current fold and save the result for future evaluation.

5 EVALUATION METHOD

5.1 Cross-validation

To measure the accuracy of a single CBR classification algorithm, we take the mean of the accuracies of each fold. The accuracy of one fold is the number of correctly assigned test instances divided by the size of the test set in this fold.

5.2 Statistical comparison of classifiers

We have 2 similarity measures, 2 reuse policies and 3 retention policies. This results in 12 possible combinations of CBR phases. We test each of the 12 combinations against three datasets, and save the accuracy results in the `evaluationResults` matrix. Each row of the matrix represents a different dataset, each column represents a different algorithm.

We use the `friedman` and `multicomp` matlab functions in order to rank the algorithms. These functions combine the Friedman test for multiple algorithm comparison with the Bonferrini-Dunn test and produce the interactive graph of the estimated algorithms' ranks with comparison intervals around them. Please remark, that the convention used by the matlab functions is different than the convention used in the Janes Demsar's paper. In matlab a higher rank signifies the better performance, whereas in the paper the lower rank indicates the better performance.

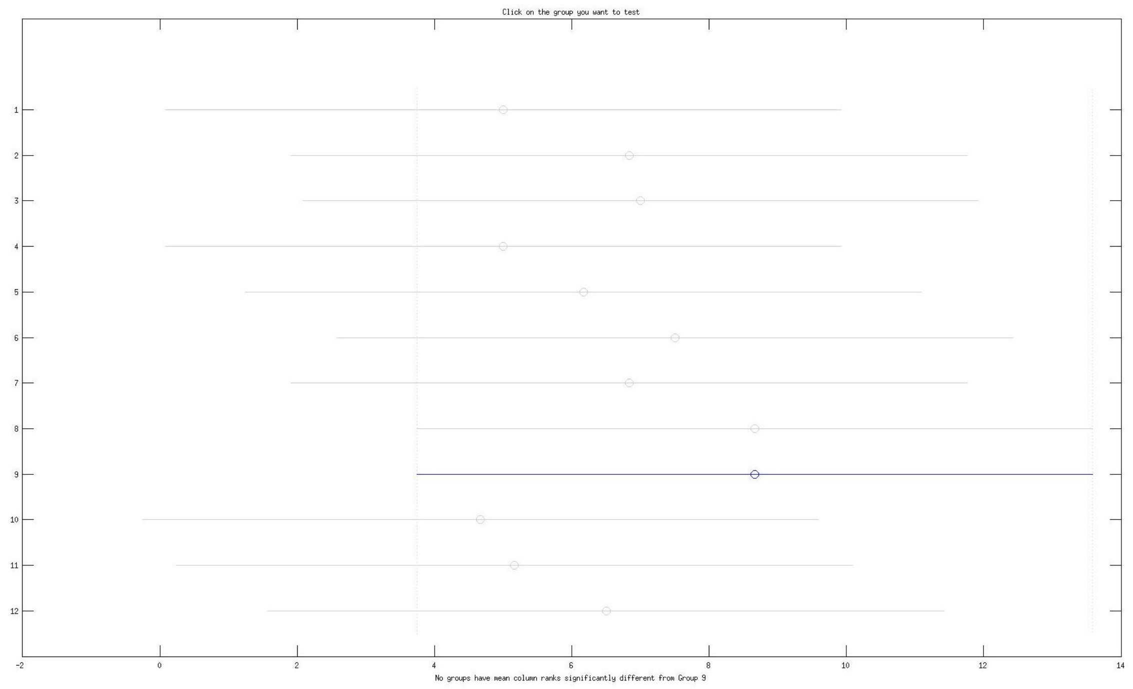
6 STANDARD CBR ALGORITHM

As we can see in the picture on the top of next page, there are two combinations that get the highest rank, though none of them is significantly different from the others.

We decided to pursue with the the following combination:

- **cosine** similarity measure;
- **closest** reuse policy;
- **none** retention policy.

From now on we will refer to this combination as the **standard** CBR algorithm.



7 FEATURE SELECTION

We use the standard algorithm to compare the aforementioned feature selection algorithms. We do not select features, instead we use the algorithms to provide the weights of particular features.

To obtain the **weighted similarity measures** we add the preprocessing step to the function calculating the cosine similarity, as the cosine similarity measure was included in the standard CBR algorithm. In the preprocessing step, we find the weights of particular features. We obtain the weights with one of the two following filter feature selection methods:

- **Mutual Information Maximization** (`http://www.cs.man.ac.uk/~gbrown/fstoolbox/`);
- **ReliefF algorithm** (`http://www.mathworks.es/es/help/stats/relieff.html`).

It turns out that the later feature selection algorithm provides better results for all three datasets. Surprisingly however, the results obtained with the feature selection are worse than those obtained without it. We also observed, that the best combination of phases without feature weighting (i.e. the standard algorithm) doesn't necessarily perform best if we apply feature selection.