

Introduction to Machine Learning

Work 2

Adaptive Case-based reasoning exercise

Contents

1	Adaptive Case-based Reasoning exercise	2
1.1	Introduction	2
1.2	Methodology of the analysis	2
1.3	Work to deliver	5
2	Data sets	7

1 Adaptive Case-based Reasoning exercise

1.1 Introduction

In this exercise you will learn about case-based reasoning and maintenance with an adaptive case-based reasoning implementation. You will apply these techniques to a classification task. It is assumed that you are familiar with the concept of cross-validation. If not, you can read this paper:

[1] R. Kohavi. *A study of cross-validation and bootstrap for accuracy estimation and model selection*. In *Proceedings of the International Joint Conferences on Artificial Intelligence IJCAI-95*. 1995.

Briefly, an s-fold cross validation ($s = 10$ in your case) divides a data set into s equal-size subsets. Each subset is used in turn as a test set with the remaining (s-1) data sets used for training.

For the validation of the different algorithms, you need to use a T-Test or another statistical method. Next reference is a **reading proposal** on this topic:

[2] Janez Demšar. 2006. *Statistical Comparisons of Classifiers over Multiple Data Sets*. *J. Mach. Learn. Res.* 7 (December 2006), 1-30.

This article details how to compare two or more learning algorithms with multiple data sets.

1.2 Methodology of the analysis

As in the previous work assignment, you will analyze the behavior of the different algorithms by comparing the results in a pair of well-known data sets (medium and large size) from the UCI repository. In that case, you will also use the class as we are testing several supervised learning algorithms. In particular, in this exercise, you will receive the data sets defined in .arff format but divided in ten training and test sets (they are the *10-fold cross-validation* sets you will use for this exercise).

This work is divided in several steps:

1. Improve the parser developed in Work 1 in order to use the class attribute, too. You should write a function in MatLab called `parser_arff_file('filename.arff')` that saves the information from a training or testing file in a `TrainMatrix` or `TestMatrix`. You need to normalize all the numerical attributes in the range [0..1]. Next you have an example of how to normalize one attribute of your data and how to get your original data back:

```
bla = 100.*randn(1,10)

minVal = min(bla);
maxVal = max(bla);

norm_data = (bla - minVal) / ( maxVal - minVal )
```

```
your_original_data = minVal + norm_data.*(maxVal - minVal)
```

Additionally, write a MatLab function that automatically repeats the process described in previous step for the 10-fold cross-validation files. That is, read automatically each training case and run each one of the test cases in the selected classifier.

2. Write two MatLab functions for maintaining the case-base of the training set `TrainMatrix`. Functions will be called `maintenanceAlgorithmX(...)`. You decide the parameters for these maintenance algorithms. Each maintenance function will call one case-base maintenance algorithm that you will implement. The simplest and also the most applied ones are those called selection based data reduction methods, see reference [4]. Obviously, you can implement others if you are more interested. **Justify your implementation and add all the references you have considered for your decisions.** An interesting paper that details quite well the CBR cycle is:

[3] Retrieval, reuse, revision and retention in case-based reasoning *The Knowledge Engineering Review*, Vol. 20, No. 03. (2005), pp. 215-240, doi:10.1017/S0269888906000646 by Ramon de Mantaras, David Mcsherry, Derek Bridge, et al.

A paper that summarizes all the approaches that can be used to perform case-based maintenance in CBR is:

[4] Overview of Maintenance for Case-Based Reasoning Systems. Abir Smiti and Zied Elouedi. *International Journal of Computer Applications*, Vol. 32, No. 2, 2011

At the end, you will have two case-based maintenance algorithms that reduce the size of the original case base. You should analyze the behavior of these two functions in terms of size reduction of the case base and performance of the algorithm. Extract conclusions by analyzing **two large** enough data sets. At least one of these data sets will contain numerical and nominal data.

3. Write a MatLab function for classifying each instance from the `TestMatrix` using the `TrainMatrix` to a classifier called `acbrAlgorithm(...)` (see reference [5]). You decide the parameters for this classifier and for each one of the phases of the algorithm. This ACBR function will call each one of the phases of the ACBR. Write a function for each one of the phases of the ACBR. **Justify your implementation and add all the references you have considered for your decisions.** The paper that details the ACBR cycle is:

[5] Adaptive case-based reasoning using retention and forgetting strategies. *Knowledge-Based Systems*. 24, 2 (March 2011), 230-247 by M. Salamó and M. López-Sánchez. 2011. DOI=10.1016/j.knosys.2010.08.003 <http://dx.doi.org/10.1016/j.knosys.2010.08.003>

- a. For the retrieval, `acbrRetrievalPhase(...)` function, you should consider the Euclidean distance. Adapt the Euclidean distance to handle all kind of attributes. Assume that the retrieval phase returns the K most similar instances (i.e., also known as cases) from the `TrainMatrix` to the `current_instance`. The value of K will be setup in your evaluation to 3, 5, and 7.

- b. For the reuse, `acbrReusePhase(...)`, you may consider to use the most similar retrieved case or alternatively to use a voting policy. The resulting case will be used to provide a solution to the `current_instance`.
- c. For the revision, `acbrRevisionPhase(...)`, you do not need to implement it for the ACBR purpose. The revision phase will serve you for analyzing if the algorithm has correctly solved the new problem (i.e., `current_instance`). It will be used for the evaluation of the algorithm. In this phase you can store your results in a memory data structure or in a file.
- d. For the review phase, `acbrReviewPhase(...)`, you will implement the update goodness measure and the oblivion by goodness forgetting strategy used in [5].
- e. For retaining, `acbrRetentionPhase(...)`, you will implement two of the policies described in [5]: DD, DE, MG or LE. Additionally, you will implement two opposite retention strategies: NR (never retain), which does not change the case base, and AR (always retain), which increases its case base with the new testing cases. The last policy retains all the new solved cases, and you will store the solution with the real one. These two approaches will be called `acbrNoRetentionPhase()` and `acbrAlwaysRetentionPhase()`.

At the end, you will have ACBR with a retrieval function (Euclidean), a reuse function (that you decide), a review function and four retention policies (2 simple ones and 2 selected from [5]). You should analyze the behavior of these functions in the ACBR algorithm and decide which combination results in the best ACBR algorithm. Notice that you can analyze separately the retention and the oblivion so that you will obtain several combinations of your ACBR algorithm. For example, if you have chosen DD strategy you can also analyze DD-O (that is DD with oblivion). You can compare your results in terms of classification accuracy, efficiency, and final case-base size. Extract conclusions by analyzing **two large** enough data sets (the same you used before in step 2). At least one of these data sets will contain numerical and nominal data.

Additionally, compare the case-based maintenance algorithms from step 2 to the adaptive and online maintenance performed by the ACBR algorithm.

Moreover, test what happens when the case base received by the ACBR algorithm has been preprocessed by one of the case-base maintenance algorithms implemented in step 2.

4. (OPTIONAL STEP). Modify the retrieval similarity function in the CBR so that it implements a weighted function called `weightedACBRRetrievalPhase()`. Each weight denotes if an attribute is considered or not for the similarity. A weight value of 1.0 denotes that the attribute will be used by the similarity. By contrast, a weight value of 0.0 shows that the attribute is useless and it is not going to be used.

The weights can be extracted using a weighting metric or a feature selection algorithm. You may choose **two algorithms** (filter or wrapper, as you wish). Use them as a preprocessing step. For example, you can use ReliefF, Information Gain, or the Correlation, among others. There are several MatLab toolboxes that also include most of the well-known metrics for feature selection algorithms. You can use the implementations that exist in MatLab for your feature selection implementations.

Analyze the results of the `weightedACBRalgorithm` in front of the previous `acbrAlgorithm` implementation. To do it, setup both algorithms with the best combination obtained in your previous analysis (step 3). In this case, you will only analyze your results in terms of classification accuracy.

The schedule for these steps is as follows:

- Week 1. Steps 1, 2
- Week 2. Steps 3, 4

1.3 Work to deliver

In this work, you will implement two case-base maintenance algorithms and an adaptive case-based reasoning. Additionally, if you wish, you can implement an adaptive case-based reasoning with feature selection. You may select 2 data sets (large enough to extract conclusions) for your analysis. At the end, you will find a list of the data sets available.

You will use your code in MatLab to extract the performance of the different combinations. Performance will be measured in terms of classification accuracy, efficiency and case-base size. The accuracy measure is the average of correctly classified cases. That is the number of correctly classified instances divided by the total of instances in the test file. The efficiency is the average problem-solving time. For the evaluation, you will use a T-Test or another statistical method [2].

From the accuracy, efficiency and case-base size results, you will extract conclusions showing graphs of such evaluation and reasoning about the results obtained.

In your analysis, you will include several considerations.

1. You will analyze the behavior of the preprocessing case-based maintenance algorithms.
2. You will analyze the ACBR (with no weighting or selection). You will analyze which is the most suitable combination of phases for the ACBR. The one with the highest accuracy. This ACBR combination will be named as the best ACBR.
3. Once you have decided the best ACBR combination. You will analyze it in front of using a preprocessing case-base maintenance algorithm. Optionally, you can also analyze the combination of the best ACBR with two feature selection algorithms. The idea is to extract conclusions of which feature selection algorithm is the best one.

For example, some of questions that it is expected you may answer with your analysis:

- Which is the best value of K for the retrieval and (optionally) for the reuse phase?
- Did you find useful the maintenance of the case-base in a case-based reasoning algorithm?
- Which is the best maintenance policy for the ACBR?
- Did you find differences in performance among the ACBR and the preprocessing maintenance algorithms?
- (optional) Did you find differences in performance among the ACBR and the weighted ACBR?
- (optional) According to the data sets chosen, which feature selection algorithm provides you more advice for knowing the underlying information in the data set?
- (optional) In the case of the feature selection CBR, how many features were removed? Which are the features selected for each one of the feature selection algorithms?
- (optional) Which criterion have you used to decide the features that should be removed?

Apart from explaining your decisions and the results obtained, it is expected that you reason each one of these questions along your evaluation. Additionally, you should explain how to execute your code. Remember to add any reference that you have used in your decisions.

You should deliver a word or pdf document as well as the code in MatLab in Racó by December, 27th, 2015.

2 Data sets

Below, you will find a table that shows in detail the data sets that you can use in this work. All these data sets are obtained from the UCI machine learning repository. First column describes the name of the domain or data set. Next columns show #Cases = Number of cases or instances in the data set, #Num. = Number of numeric attributes, #Nom = Number of nominal attributes, #Cla. = Number of classes, Dev.Cla. = Deviation of class distribution, Maj.Cla. = Percentage of instances belonging to the majority class, Min.Cla. = Percentage of instances belonging to the minority class, MV = Percentage of values with missing values (it means the percentage of unknown values in the data set). When the columns contain a '-', it means a 0. For example, the Glass data set contains 0 nominal attributes and it is complete as it does not contain missing values.

Domain	#Cases	#Num.	#Nom.	#Cla.	Dev.Cla.	Maj.Cla.	Min.Cla.	MV
<i>Adult</i>	48,842	6	8	2	26.07%	76.07%	23.93%	0.95%
<i>Audiology</i>	226	-	69	24	6.43%	25.22%	0.44%	2.00%
<i>Autos</i>	205	15	10	6	10.25%	32.68%	1.46%	1.15%
* <i>Balance scale</i>	625	4	-	3	18.03%	46.08%	7.84%	-
* <i>Breast cancer Wisconsin</i>	699	9	-	2	20.28%	70.28%	29.72%	0.25%
* <i>Bupa</i>	345	6	-	2	7.97%	57.97%	42.03%	-
* <i>cmc</i>	1,473	2	7	3	8.26%	42.70%	22.61%	-
<i>Horse-Colic</i>	368	7	15	2	13.04%	63.04%	36.96%	23.80%
* <i>Connect-4</i>	67,557	-	42	3	23.79%	65.83%	9.55%	-
<i>Credit-A</i>	690	6	9	2	5.51%	55.51%	44.49%	0.65%
* <i>Glass</i>	214	9	-	2	12.69%	35.51%	4.21%	-
* <i>TAO-Grid</i>	1,888	2	-	2	0.00%	50.00%	50.00%	-
<i>Heart-C</i>	303	6	7	5	4.46%	54.46%	45.54%	0.17%
<i>Heart-H</i>	294	6	7	5	13.95%	63.95%	36.05%	20.46%
* <i>Heart-Statlog</i>	270	13	-	2	5.56%	55.56%	44.44%	-
<i>Hepatitis</i>	155	6	13	2	29.35%	79.35%	20.65%	6.01%
<i>Hypothyroid</i>	3,772	7	22	4	38.89%	92.29%	0.05%	5.54%
* <i>Ionosphere</i>	351	34	-	2	14.10%	64.10%	35.90%	-
* <i>Iris</i>	150	4	-	3	-	33.33%	33.33%	-
* <i>Kropt</i>	28,056	-	6	18	5.21%	16.23%	0.10%	-
* <i>Kr-vs-kp</i>	3,196	-	36	2	2.22%	52.22%	47.78%	-
<i>Labor</i>	57	8	8	2	14.91%	64.91%	35.09%	55.48%
* <i>Lymph</i>	148	3	15	4	23.47%	54.73%	1.35%	-
<i>Mushroom</i>	8,124	-	22	2	1.80%	51.80%	48.20%	1.38%
* <i>Mx</i>	2,048	-	11	2	0.00%	50.00%	50.00%	-
* <i>Nursery</i>	12,960	-	8	5	15.33%	33.33%	0.02%	-
* <i>Pen-based</i>	10,992	16	-	10	0.40%	10.41%	9.60%	-
* <i>Pima-Diabetes</i>	768	8	-	2	15.10%	65.10%	34.90%	-
* <i>SatImage</i>	6,435	36	-	6	6.19%	23.82%	9.73%	-
* <i>Segment</i>	2,310	19	-	7	0.00%	14.29%	14.29%	-
<i>Sick</i>	3,772	7	22	2	43.88%	93.88%	6.12%	5.54%
* <i>Sonar</i>	208	60	-	2	3.37%	53.37%	46.63%	-
<i>Soybean</i>	683	-	35	19	4.31%	13.47%	1.17%	9.78%
* <i>Splice</i>	3,190	-	60	3	13.12%	51.88%	24.04%	-
* <i>Vehicle</i>	946	18	-	4	0.89%	25.77%	23.52%	-
<i>Vote</i>	435	-	16	2	11.38%	61.38%	38.62%	5.63%
* <i>Vowel</i>	990	10	3	11	0.00%	9.09%	9.09%	-
* <i>Waveform</i>	5,000	40	-	3	0.36%	33.84%	33.06%	-
* <i>Wine</i>	178	13	-	3	5.28%	39.89%	26.97%	-
* <i>Zoo</i>	101	1	16	7	11.82%	40.59%	3.96%	-