Journal of Machine Learning Research 9 (2008) 2695-2698 Submitted 5/08; Revised 9/08; Published 12/08

JNCC2: The Java Implementation Of Naive Credal Classifier 2

Giorgio Corani GIORGIO@IDSIA.CH
Marco Zaffalon ZAFFALON@IDSIA.CH
IDSIA
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
CH-6928 Manno (Lugano), Switzerland

Editor: Mikio Braun

Abstract

JNCC2 implements the *naive credal classifier 2* (NCC2). This is an extension of naive Bayes to imprecise probabilities that aims at delivering robust classifications also when dealing with small or incomplete data sets. Robustness is achieved by delivering set-valued classifications (that is, returning multiple classes) on the instances for which (i) the learning set is not informative enough to smooth the effect of choice of the prior density or (ii) the uncertainty arising from missing data prevents the reliable indication of a single class. JNCC2 is released under the GNU GPL license.

Keywords: imprecise probabilities, missing data, naive Bayes, naive credal classifier 2, Java

1. Introduction

JNCC2 is the Java implementation of *naive credal classifier 2* (NCC2) (Corani and Zaffalon, 2008). NCC2 extends naive Bayes (NBC) to *imprecise probabilities* (Walley, 1991) in order to deliver reliable classifications even on small or incomplete data sets.

A problem of NBC is that, on small data sets, it may return *prior-dependent* classifications, that is, it might identify a different class as the most probable one, depending on the prior density adopted to infer the classifier. In some cases this can lead NBC to issue fragile predictions. To deal with this problem, NCC2 specifies a set of prior densities, referred to as *prior credal set*; the credal set is then turned into a set of posteriors via element-wise application of Bayes' rule. Eventually, NCC2 returns the classes that are *non-dominated* with respect to the set of posterior densities (class c_1 dominates class c_2 if the probability of c_1 is larger than the probability of c_2 for *all* the posteriors). When faced with an instance that would be classified in a prior-dependent way by naive Bayes, NCC2 will detect multiple non-dominated classes and will then return multiple classes; this is an *indeterminate classification*.

As for missing data, NCC2 assumes that the missingness process (MP) which generates missing data can be either *MAR* (that is, missing at random), or unknown; in the latter case, it is referred to as non-MAR. As MAR missing data can be safely ignored (Little and Rubin, 1987), NCC2 ignores them. On the other hand, NCC2 deals *conservatively* with non-MAR missing data, that is, it considers all the possible replacements for non-MAR missing data. NCC2 can handle mixed situations where some features are subject to a MAR MP and some others to a non-MAR MP; moreover, the list of features subject to the MAR and to the unknown MP can be different between training and test set. The conservative treatment of non-MAR missing data generates additional indeterminacy of NCC2, as a way to preserve reliability despite the information hidden by missing values and by the fact that the MP is unknown.

NCC2 can hence be seen as separating “easy” instances, over which it returns a single class, from “hard” instances, over which it returns an indeterminate classification. Experimental evaluations have shown that the accuracy of naive Bayes sharply drops on the hard instances, while on the same instances NCC2 remains reliable thanks to the indeterminate classifications.

Programming language: Java.
Developer: Giorgio Corani (IDSIA, Switzerland).
Open source license: GNU GPL.
Website: `www.idsia.ch/~giorgio/jncc2.html`.
Software required: Java Runtime Environment 5.0 or higher.
Operating system: OS independent.
User interface: command-line.

Figure 1: Essential information about JNCC2.

The zip file downloadable from the JNCC2 website contains executables, sources, examples, user manual and tutorial. A GUI version of the software will be released in the near future and will be published on the same website.

2. Indicators of Performance

The performance of NBC is measured by the *accuracy*, that is, the percentage of correct classifications.

The performance of NCC2 is measured by several indicators: *determinacy*: the percentage of classifications having as output a unique class; *single accuracy*: the accuracy of NCC2 when it is determinate; *indeterminate output size*: the average number of classes returned when NCC2 is indeterminate; *set-accuracy*: the percentage of indeterminate classifications that contain the actual class.

To assess the effectiveness of the approach based on imprecise probabilities, the accuracy of naive Bayes is moreover measured separately on the instances recognized as hard and easy by NCC2. If NCC2 is effective at separating easy from hard instances, a significant difference will be found between the two measures.

Moreover, JNCC2 computes the confusion matrices of NBC and NCC2; in case of NCC2 the confusion matrix refers to the determinate classifications only.

3. Some Implementation Details

JNCC2 loads data from ARFF files; this is a plain text format, originally developed for WEKA (Witten and Frank, 2005). A large number of ARFF data sets, including the data sets from the UCI repository, is available from the address `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`.

As a pre-processing step, JNCC2 discretizes all the numerical features, using the supervised discretization algorithm of Fayyad and Irani (1993). The discretization intervals are computed on the training set, and then applied unchanged on the test set.

NCC2 is implemented exploiting the computationally efficient procedure described in (Corani and Zaffalon, 2008, Appendix A).

Algorithm 1 Pseudocode for validation via testing file.

```
validateTestFile()

  /*loads training and test file; reads list of non-Mar features; discretizes features*/
  parseArffFile();
  parseArffTestingFile();
  parseNonMar();
  discretizeNumFeatures();

  /*learns and validates NBC*/
  nbc = new NaiveBayes(trainingSet);
  nbc.classifyInstances(testSet);

  /*learns and validates NCC2; the list of non-Mar features in training and testing is required*/
  ncc2 = new NaiveCredalClassifier2(trainingSet, nonMarTraining, nonMarTesting);
  ncc2.classifyInstances(testingSet);

  /*writes output files*/
  writePerfIndicators();
  writePredictions();
```

JNCC2 can perform three kinds of experiments: training and testing, cross-validation, and classification of instances of the test set whose class is unknown. The pseudo code of the experiment with training and testing is described by Algorithm 1.

4. Examples

To run the following examples, move to the directory `examples/completeData`, generated under the JNCC2 directory after unzipping the package. To perform a training and testing experiment, type for instance:

“`java jncc20.Jncc .  iris.training.arff iris.testing.arff`”.

As a consequence, JNCC2 will load the training and test set, discretize the numerical features, learn both NBC and NCC2, and use them to predict the instances of the test set. Then it will write the performance measures and the predictions to file. Similar experiments can be performed also with the glass and contact-lenses data sets, provided in the same directory.

To run a cross-validation experiment, type for instance:

“`java jncc20.Jncc .  iris.training.arff cv`”.

JNCC2 will perform 10 runs of 10-folds stratified cross-validation, that is, 100 training/test experiments. JNCC2 will report the performance indicators to file, together with their observed standard deviations, but it will not write the predictions. (As a side remark, if one wants to run cross-validation, there is no need of splitting the original data set into a training and a testing file, as it is has been done in this directory.)

The directory `examples/missingData` contains two examples of data sets containing missing data; a look at the provided files `NonMar.txt` should make it clear how to declare the non-MAR features.

The directory `examples/unkClasses` contains two examples in which the class of the instances of the testing set is not available. For the iris data set, the experiment is for instance started as follows:

“`java jncc20.Jncc . iris.training.arff iris.testingUnkClasses.arff unknownclasses`”.

Acknowledgments

We are grateful to all the reviewers for their valuable comments. Work for this paper has been partially supported by the Swiss NSF grants 200021-113820/1 and 200020-116674/1, and by the Hasler Foundation (Hasler Stiftung) grant 2233.

References

G. Corani and M. Zaffalon. Learning reliable classifiers from small or incomplete data sets: the naive credal classifier 2. *Journal of Machine Learning Research*, 9:581–621, 2008.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, San Francisco, CA, 1993. Morgan Kaufmann.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley, New York, 1987.

P. Walley. *Statistical Reasoning With Imprecise Probabilities*. Chapman and Hall, New York, 1991.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (2nd Edition)*. Morgan Kaufmann, 2005.