An Object-oriented Library for Systematic Training and Comparison of Classifiers for Computer-assisted Tumor Diagnosis from MRSI Measurements

Frederik O. Kaster[1,2], Stephan Kassemeyer[1], Bernd Merkel[3], Oliver Nix[2], Fred A. Hamprecht[1]

[1]Ruprecht-Karls-Universität, Heidelberg
[2]Deutsches Krebsforschungszentrum, Heidelberg
[3]Fraunhofer MEVIS, Institut für Bildgestützte Medizin, Bremen
frederik.kaster@iwr.uni-heidelberg.de

Abstract. We present an object-oriented library for the systematic training, testing and benchmarking of classification algorithms for computer-assisted diagnosis tasks, with a focus on tumor probability estimation from magnetic resonance spectroscopy imaging (MRSI) measurements. In connection with a graphical user interface for data annotation, it allows clinical end users to flexibly adapt these classifiers towards changed classification tasks and to perform quality control of the results. This poses an advantage over previous classification software solutions, which had to be manually adapted by pattern recognition experts for every change in the data acquisition protocols. In this article, we concentrate on software architecture and design principles of this library.

1 Introduction

In the past years, pattern recognition methods have proved their efficacy for the automated tumor detection based on MRSI measurements [1]. They often show higher robustness against measurement artifacts and noise than tumor detection methods that depend on the explicit quantitation of relevant metabolites. However, the use of quantitation-based algorithms in the clinic is already well-established, since they are typically included in the software packages supplied by MR scanner manufacturers. Furthermore, several stand-alone software products such as LCModel [2], jMRUI [3] or MIDAS [4] exist. In contrast, the application of pattern recognition-based methods still has to gain ground in clinical routine, and existing software products such as CLARET [5] or HealthAgents [6] are more of prototypical character. We believe that this may be partially due to differences in the flexibility with which both categories of algorithms can be adjusted to different experimental conditions (e.g., changes in scanner hardware and in measurement protocols) or to a different imaged organ. For quantitation-based methods one must only update the metabolite basis spectra to a given experimental setting, which can be achieved by quantum-mechanical simulation. On

the other hand, methods based on pattern recognition achieve their robustness by foregoing an explicit data model. Instead they learn the mapping from observations to tumor class labels from training examples provided by human expert annotators, which have to be provided anew for every change in experimental conditions. Since there exist many different classification techniques whose relative and absolute performance on a given task cannot be predicted beforehand, for every change in conditions a benchmarking experiment as in [1] should also be conducted to select the best classifier and monitor the classification quality.

While we cannot obviate the need for classifier retraining, benchmarking and quality assessment, we have designed an object-oriented C++ library which assists this task better than existing software. Extensibility was an important design criterion: by providing abstract interfaces for classifiers, data preprocessing procedures and evaluation statistics, user-defined classes may be plugged in with moderate effort. Hereby it follows similar ideas as general purpose classification frameworks such as Weka (http://www.cs.waikato.ac.nz/ml/weka/), TunedIT (http://tunedit.org/) or RapidMiner (http://www.rapid-i.com). However, it is much more focused in scope and tailored towards medical diagnostic applications. In contrast to an earlier conference contribution [7] which focused on the validation results on prostate MRSI measurements acquired at 1.5 Tesla, this paper outlines the software design architecture.

2 Materials and Methods

The library is designed for the following use case: the user defines several classifiers to be compared, the free classifier-specific parameters to be adjusted in parameter optimization and custom preprocessing steps for the data. A training and test suite is then defined, which may contain several classification tasks, e.g. predicting the voxel class (tumor vs. healthy) and the signal quality (good vs. poor). Every classifier is assigned to a preprocessing pipeline, which transforms the observations into training and test features. Some elements of this pipeline may be shared across several classifiers, while others may be specific for one classifier. Input data (observations and labels) are passed, preprocessed and partitioned into cross validation folds, and the parameters of every classifier are optimized on one fold by maximizing an estimate for the generalization error. The mean and variance of all evaluation statistics are then estimated using cross-validation, and statistical tests are conducted to decide whether the classifiers differ significantly in performance. Typically not only two, but multiple classifiers are compared against each other, which must be considered when judging significance: e.g. for five classifiers with equal performance, we have ten pairwise comparisons and a significant difference ($p_{\text{raw}} < 0.05$) is expected to occur with a probability of $1 - 0.95^{10} \approx 0.40$. Hence the "raw" p-values from statistical tests must be corrected accordingly. Finally the classifiers are retrained on the total data for predicting the class of unlabeled examples. Trained classifiers and test results may then be saved and reloaded for persistence.

3 Results

We developed a C++ library for managing multiple classifiers, preprocessing procedures and evaluation statistics: abstract base classes for all these entities provide flexibility for later adjustments. The class architecture is depicted in Fig. 1. Concerning the classifiers, we implemented bindings for support vector machines, random forests, ridge regression and principal components regression. External libraries such as LIBSVM (http://www.csie.ntu.edu.tw/ cjlin/libsvm/) or VIGRA (http://hci.iwr.uni-heidelberg.de/vigra/) are linked to provide the underlying classification algorithms. As exemplary preprocessing and feature extraction steps, we implemented a MRSI specific preprocessor that performs e.g. the extraction of relevant parts from the spectrum and that is contained in all pipelines, a whitening transformation mainly for use with SVMs and a singular value decomposition transformation for use with regression-based classifiers. Possible evaluation statistics are e.g. precision, recall and ROC curves. For p-value-adjustment, we provide McNemar's test and a recently proposed adjusted t-test [8] with multiple comparison adjustment of the p-values by Holm's step-down or Hochberg's step-up method as in [9].

All classifiers, all preprocessors and all statistics pertaining to a certain classifier are controlled by a dedicated manager object, which enforces consistency. Furthermore, every classifier encapsulates a parameter manager object controlling which parameter combinations are tested during parameter optimization.

Data input and output for training, testing and storing of the classifiers can be customized by passing user-defined input and output function objects; we use HDF5 as main storage format. For integration into a user interface, other function objects may be passed that can report progress or status information or listen for abort requests at regular checkpoints.

In order to further aid the clinical users in spectrum annotation, we developed a graphical user interface in MeVisLab (http://www.mevislab.de) that displays MRSI spectra from a selected slice in the context of its neighbor spectra, which can then be labeled on an ordinal scale and imported into the classification library. Hence the classifier can be flexibly adapted to new experimental protocols: Clinical end users only need to load the training data sets, manually annotate the spectra in the GUI, save the annotations and start a training and testing suite. Since in practice many spectra are not evaluable owing to excessive noise or the presence of artifacts, we enable the users to define two independent labels for each spectrum, which encode voxel class and signal quality. By training two independent classifiers for both tasks (as in [5]), it is afterwards possible to judge the reliability of the automated classification of new examples.

The library was validated with 1.5 Tesla MRSI data of prostate carcinomas (both for voxel class and signal quality classification). The outcome was already discussed in [7]: for signal quality prediction with 36864 training examples, correct classification rates (CCR) of 96.5 % – 97.3 % and area under the ROC curve values of 98.9 % – 99.3 % could be achieved by the different classifiers; and for voxel class prediction with 2746 training examples, CCRs of 90.9 % – 93.7 % and AUCs of 95 % – 98 % were obtained.

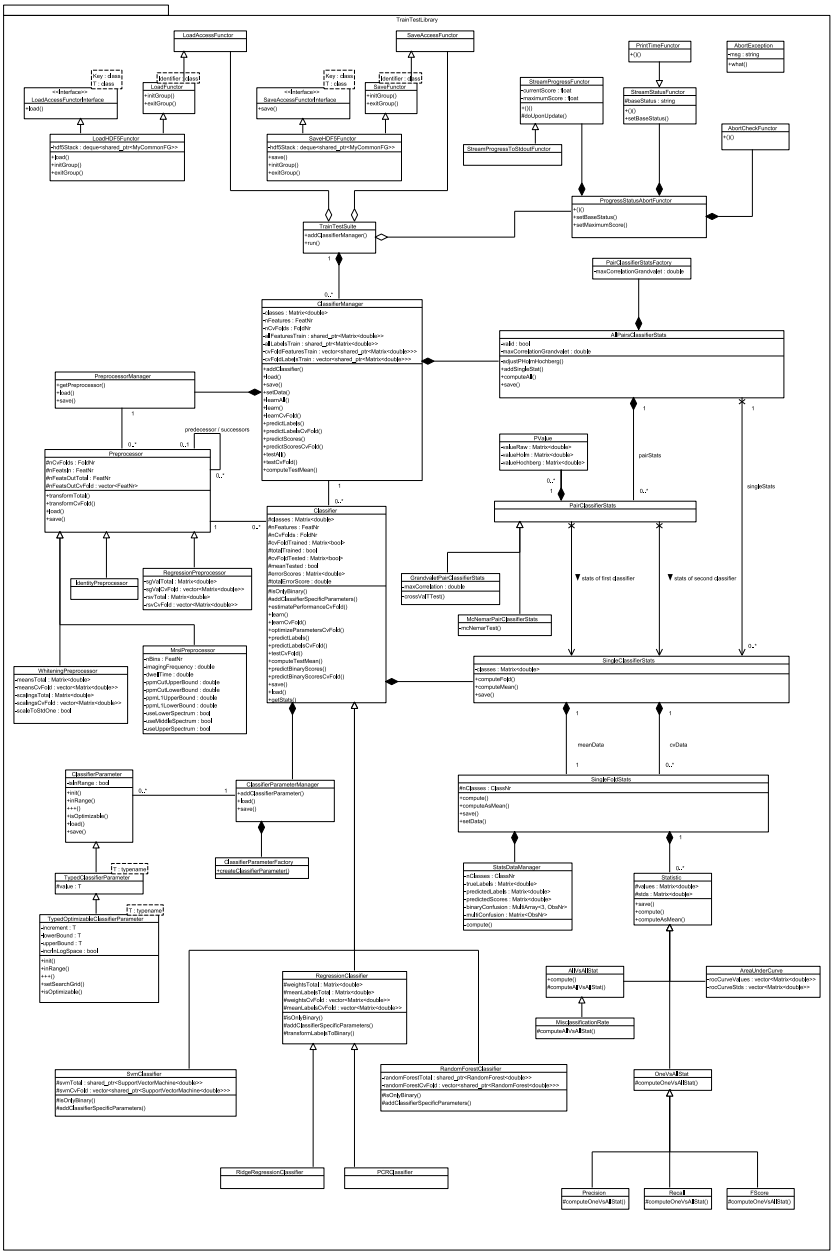

Fig. 1. Simplified UML class diagram of our library. Note that it comprises functionality for classification and managing the associated parameters (lower left), preprocessing and feature extraction (left), data input and output (top), and evaluation statistics (right and bottom right). The functors for loading and saving data, and streaming progress and status information (top part) allow the integration into user interfaces. We recommend to view this figure in the electronic version of the article.

4 Discussion

To our best knowledge, this is the first C++ library specifically designed for medical applications which allows principled comparison of classifier performance and significance testing. We believe that this will help automated quality assessment and the conduction of clinical studies. This statistical characterization functionality sets our software apart from e.g. HealthAgents [6] or the system in [10].

The adaptation to 3 Tesla MRSI measurements of brain and prostate carcinomas is ongoing. Furthermore this software will eventually be integrated into the RONDO software platform for integrated tumor diagnostic and radiotherapy planning (http://www.projekt-dot-mobi.de). Stringent quality assessment will be required for clinical application: the software has to be tested and reviewed extensively, and the correctness of the spectral annotations must be confirmed via histopathological assessments by several independent pathologists as in [1].

Acknowledgement. We acknowledge fruitful discussions with Ralf Floca and Ullrich Koethe. The graphical user interface was initiated by Markus Harz. This research was supported by the Helmholtz International Graduate School for Cancer Research and by the BMBF within the DOT-MOBI project (grant no. 01IB08002).

References

1. García-Gomez J, Luts J, Julià-Sapé M, et al. Multiproject-multicenter evaluation of automatic brain tumor classification by magnetic resonance spectroscopy. Magn Reson Mater Phy. 2009;22:5–18.
2. Provencher S. Automatic quantitation of localized in vivo ^{1}H spectra with LCModel. NMR Biomed. 2001;14(4):260–4.
3. Stefan D, Cesare FD, Andrasescu A, et al. Quantitation of magnetic resonance spectroscopy signals: the jMRUI software package. Meas Sci Technol. 2009;20:104035.
4. Maudsley A, Darkazanli A, Alger J, et al. Comprehensive processing, display and analysis for *in vivo* MR spectroscopic imaging. NMR Biomed. 2006;19:492–503.
5. Kelm B, Menze B, Neff T, et al. CLARET: a tool for fully automated evaluation of MRSI with pattern recognition methods. Proc BVM. 2006; p. 51–5.
6. González-Vélez H, Mier M, Julià-Sapé M, et al. Health agents: distributed multiagent brain tumor diagnosis and prognosis. Appl Intell. 2009;30:191–202.
7. Kaster F, Kelm B, Zechmann C, et al. Classification of spectroscopic images in the DIROlab environment. In: Proc IFMBE. vol. 25/V; 2009. p. 252–5.
8. Grandvalet Y, Bengio Y. Hypothesis testing for cross-validation. Département d'Informatique et Recherche Opérationelle, University of Montréal; 2006. TR 1285.
9. Demšar J. Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res. 2006 Dec;7:1–30.
10. Neuter BD, Luts J, Vanhamme L, et al. Java-based framework for processing and displaying short-echo-time magnetic resonance spectroscopy signals. Comput Methods Programs Biomed. 2007;85:129–37.