Francois Germain – Slide Presentation III – Thursday 22 October 2009

ACE COMPONENT

Summary

Introduction

The ACE (Autonomous Classification Engine) component is a framework for the use and optimization of classifiers. It is one of the different elements composing jMIR, an open-source software suite implemented in Java for use in music information retrieval (MIR) research.

Presentation of ACE

The ACE framework experiments a variety of classifiers, classifier ensembles and dimensional reduction techniques. It is designed to facilitate and optimize classification tasks for researchers in music information research (MIR).

The ACE 1.1 release is a command-line software. It allows the user to perform three tasks. The first one, implemented in the ACEFileConverter class, is the translation of ACE XML files in Weka ARFF files. The second one, implemented in the InstanceLabeller class, is the training of classifiers using an input training set and the taxonomy. Finally, the third one, implemented in the ClassificationTester class, is the classification of an input data set using a trained classifier and a taxonomy. The ACE 2.0 is currently being released and it offers a GUI interface to the user.

Limits of existing machine learning frameworks

The ACE implementation is inspired by the weaknesses of existing machine learning frameworks for pattern recognition in MIR research context (McKay et al. 2005).

Firstly, the general frameworks, which were not designed specifically for music research, are compared. A first example is PRTools (Van der Heijden et al. 2004), a MATLAB toolbox. The problem is that it is dependent of proprietary software, and it is not redistributable. The second one is Weka (Witten and Frank 2000). It is an open-source software implemented in Java. It fits most of the objectives drawn by Cory McKay as necessary for MIR research. That's why the ACE engine is based on machine learning algorithms implemented in Weka. The main issues of this framework are related to the file format used, The Weka ARFF format, which is too rigid for applications in music.

The music dedicated frameworks have also some weaknesses. For an example, Marsyas (Tzanetakis and Cook 1999) can perform classification tasks but it was designed as a feature extractor rather than a classification framework. Furthermore, it can only perform audio classification and doesn't include MIDI functionality. Another problem is that it is implemented in C++ and so has a limited portability compared to Java softwares. Another example is M2K (Music-to-knowledge), a framework based on D2K (Downie 2004). It suffers from license issues that make it non suitable for a research purpose.

Technical aspects of ACE

In order to have more flexibility with the exhange file formats in jMIR, Cory McKay designed a specific format. It solves the issues related to the ARFF format. Indeed, these files have some problematic constraints. For an example, each instance can only be assigned to one class. So it is impossible to specify two genres for one single feature. It is also impossible to keep tracks of the relations between the samples if they are coming from the same song or the same database. No hierarchy can be specified between the different labels. These constraints are problematic in the case of MIR research. The ACE XML file format solves these issues. The choice of XML is due to the high standardization of the format, and the fact that XML files are verbose makes them easier to use for debugging. Furthermore, the format allows the storage of metadata to strenghten the independence between the feature extraction and the classification. Cory McKay chose to divide each output in many files, each dedicated to a specific kind of data (features, feature definitions, classificatier parameters…).

The ACE framework allows the use of many classical classification techniques. It includes feedforward neural networks, support vector machines, nearest neighbour classifiers, decision tree classifiers, and bayesian classifiers.

On contrary to the machine learning frameworks, ACE implements meta-learning. That means that it exploits information from classification training to optimize classification. For this purpose, it combines the answer from several classification models. This approach has many advantages (Dietterich 2000). It has an obvious statistical performance, but it can also significantly improve the results in the case of local optima methods, or in the case of non adapted classifiers. The different methods used in ACE are voting, dynamic selection, stacking, bagging, and boosting. This last technique and its derived version called AdaBoost (Freund and Schapire 1996) are particularly efficient in experiments.

Another optimization of classification implemented in ACE is the feature dimension reduction. It applies an iterative evaluation of the feature weighting techniques. ACE uses techniques such as genetic algorithms, principal component analysis, tree search, or forward-backward algorithm.

ACE 2.0 and future work

The ACE 2.0 release implements several improvements compared to the previous version.

Firstly, the class architecture was restructured and extended to facilitate integration with other softwares and provide new fonctions for users. Secondly, the cross-validation was redesigned. Formerly, the Weka function was used but now a specific ACE class implements this functionality. Thirdly, the ACE XML Zip and Project files were designed to facilitate the use of related ACE XML files. Finally, the interface was improved, with a graphical user interface (GUI) for editing the ACE XML files, and new command-line arguments.

The future work on ACE is mainly focused on the completion of its present features. The GUI would be improved to completely replace the command-line interface. More classification algorithms, such as hidden Markov models, or recurrent neural networks, would be included. Finally, an important needed feature is the work load distribution which would improve computation time.

Conclusion

The ACE framework provides to MIR researchers a user-friendly and very complete software for classification purposes. It is designed to allow the user to test his/her own classifiers, classifier ensembles or data reduction techniques. It encourages experimentation in classification, and furthermore draws the advantages of some techniques of optimized classification.

References

Dietterich, T. G. 2000. Ensemble methods in machine learning. In *Multiple classifier systems*, J. Kittler and F. Roli eds. New York: Springer.

Downie, J. S. 2004. International music information retrieval systems evaluation laboratory (IMIRSEL): Introducing D2K and M2K. *Demo Handout at the International Conference on Music Information Retrieval*.

Fiebrink, R., C. McKay, and I. Fujinaga. 2005. Combining D2K and JGAP for efficient feature weighting for classification tasks in music information retrieval. *Proceedings of the International Conference on Music Information Retrieval*.

Freund, Y., and R. E. Schapire. 1996. Experiments with a new boosting algorithm. *Proceedings of the International Conference on Machine Learning*: 148–56.

van der Heijden, F., R. Duin, D. de Ridder, and D. Tax. 2004. *Classification, parameter estimation and state estimation - an engineering approach using Matlab*. New York: Miley.

Kittler, J. 2000. A framework for classifier fusion: Is it still needed? *Proceedings of the Joint International Workshops on Advances in Pattern Recognition*: 45–56.

Kotsiantis, S., and P. Pintelas. 2004. Selective voting. *Proceedings of the International Conference on Intelligent Systems Design and Applications*: 397–402.

McKay, C., R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga. 2005. ACE: a framework for optimizing music classification. *Proceedings of the International Conference on Music Information Retrieval*: 42–9.

McKay, C., D. McEnnis, R. Fiebrink, and I. Fujinaga. 2005. ACE: A general-purpose classification ensemble optimization framework. *Proceedings of the International Computer Music Conference*.

Sinyor, E., C. McKay, R. Fiebrink, D. McEnnis, and I. Fujinaga. 2005. Beatbox classification using ACE. *Proceedings of the International Conference on Music Information Retrieval*: 672–5.

Thompson, J., C. McKay, J. Burgoyne, and I. Fujinaga. 2009. Additions and improvements to the ACE 2.0 music classifier. *Proceedings of the International Society for Music Information Retrieval Conference*.

Tzanetakis, G., and P. Cook. 1999. MARSYAS: A framework for audio analysis. *Organized Sound* 4 (3): 169–75.

Witten, I., and E. Frank. 1999. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. *Proceedings of ICONIP/ ANZIIS/ANNES99 Future Directions for Intelligent Systems and Information Sciences*: 192–196.