ACE: A GENERAL-PURPOSE CLASSIFICATION ENSEMBLE OPTIMIZATION FRAMEWORK

Cory McKay
Music Technology
Faculty of Music
McGill University
cory.mckay@
mail.mcgill.ca

Daniel McEnnis
Music Technology
Faculty of Music
McGill University
daniel.mcennis@
mail.mcgill.ca

Rebecca Fiebrink
Music Technology
Faculty of Music
McGill University
rfiebrink@acm.org

Ichiro Fujinaga
Music Technology
Faculty of Music
McGill University
ich@music.mcgill.ca

ABSTRACT

This paper describes ACE, a framework for automatically finding effective classification methodologies for arbitrary supervised classification problems. ACE performs experiments with both individual classifiers and classifier ensembles in order to find the approaches best suited to particular problems. A special emphasis is placed on classifier ensembles, as they can be powerful tools, yet are currently rarely used in MIR research. In addition to evaluating various classification methodologies in terms of success rates, ACE also allows users to specify constraints on training and classification times.

The input to ACE is an arbitrary taxonomy accompanied by training feature vectors and their model classifications. ACE then outputs comparisons of the effectiveness of different classification methodologies, including information relating to feature weightings, dimensionality reduction and classifier combination techniques. The user may then select any of these configurations, after which s/he will be presented with trained classifiers that can be used to classify new feature vectors.

Although designed to be used easily with any existing feature extraction system, ACE is also packaged with MIDI and audio feature extraction sub-systems. In addition, plans are underway to make use of distributed computing in order to decrease processing times.

1. INTRODUCTION

Pattern recognition and automatic classification techniques are currently used for a wide variety of tasks in music research. Genre and composer classification, similarity-based music recommendation and intelligent interactive accompaniment are just a few of the areas where these techniques can be used.

Unfortunately, the variety and technical sophistication of the pattern recognition techniques available can make it difficult to choose the best approach to apply to a particular problem. Furthermore, there are not currently any freely available, easy-to-use and standardized pattern recognition systems intended to address the particular needs of music researchers. This requires researchers to spend time either adapting existing systems to their needs or implementing systems themselves. A standardized and usable system that experimentally finds the best classification approach for a given problem could help in terms of both increasing classification accuracy and saving researchers significant amounts of development time. The ACE (Autonomous Classification Engine) framework presented here is intended to address these issues. ACE includes implementations of a wide variety of pattern recognition techniques and provides interfaces intended specifically for the needs of music researchers. These interfaces include a powerful GUI, a command-line interface and Java classes that can be accessed by individual researchers' software. This allows ACE to be used for a variety of purposes by users with different levels of technical expertise.

The most powerful aspect of ACE is that it experimentally compares the performance of different classification algorithms on individual users' data sets in the context of the particular taxonomies and features that they are using. Data is also generated on the relative effectiveness of different methods of combining the classifiers into ensembles.

ACE also experimentally compares the effectiveness of a variety of dimensionality reduction techniques operating in conjunction with the different classification techniques. This is essential, as the performance of any classification system is inherently limited by the quality of the features on which it bases its decisions.

ACE analyzes the effectiveness of different approaches in terms of classification accuracy, training time and classification time. This allows users to experimentally determine the best set of techniques to use for their particular priorities. In addition, ACE is currently being modified to allow users to specify limits on how long the system has to arrive at a solution, with the result that ACE will initially pursue the most promising approaches, based on past experiments with similar data, and output the best approaches that it has found in the given time.

ACE may also be used directly as a classifier. Once users have selected the classifier(s) that they wish to use, whether through ACE optimization or using pre-existing knowledge, they need only provide ACE with a taxonomy, training feature vectors and ground truth. ACE then trains itself and presents users with trained classifier(s) that may be used to classify new feature vectors.

An important advantage of ACE is that it is open-source and free. ACE is implemented in Java, which means that the framework is portable and can be used on essentially any operating systems without any installation concerns related to compilation and linking.

ACE's portability is an important advantage for users with custom feature extraction systems, as it means that ACE can be easily installed and used on whatever platforms user's existing systems already use. ACE also addresses the needs of users without pre-existing feature

extraction systems, as it is packaged with both audio and MIDI default feature extractors.

ACE has been designed with a modular and extensible design philosophy. It is a simple and well-documented matter for users to add implementations of new classifiers, ensemble combination techniques, features or feature weighting techniques to ACE.

It should be noted that ACE can in fact be used for arbitrary classification optimization tasks. Although the interface and feature selection sub-systems are designed with the specific needs of music researchers in mind, there is nothing about the underlying implementation of ACE that limits it to musical applications.

2. RELATED RESEARCH

There has been a great deal of material published on machine learning, pattern recognition and classification. The books of Alpaydin [1], Duda, Hart and Stork [4] and Russell and Norvig [12] provide particularly good complementary overviews. Kuncheva's book [7] provides a general reference on classifier ensembles.

Despite the continuing improvement of individual classifiers, it has been argued that approaches using feature selection and classifier ensembles are still the most effective [6]. These theoretical arguments are supported by numerous experiments showing the improvements in classification accuracy brought about by using classifier ensembles (e.g. [9]). This evidence has motivated the inclusion of classifiers ensembles in ACE, something currently rare in the MIR field.

A previously published system that applies such an experimental approach to music classification is the EDS developed by Sony CSL [10]. Although Sony's system is certainly powerful, and does offer the advantages of dynamically constructing features and of allowing the use of specifiable heuristics, ACE is a more general and open system. Sony's approach focuses on genetic searches, whereas ACE experiments with a wider range of feature weighting algorithms. ACE also utilizes a wider range of classifiers including classifier ensembles. ACE also has the important advantage of being open source, and is available to the public for use and development, whereas Sony's EDS is proprietary. In addition, Sony's system has been designed with the particular needs of similarity and recommendation in mind, whereas ACE has been designed to be applicable to arbitrary classification tasks, including symbolic as well as audio classification.

3. CLASSIFIERS

ACE makes use of the Weka Java-based data mining package [14]. Weka has the advantages of full Java portability and of a licence allowing free open-source distribution. The inclusion of Weka compatibility means that all classification methodologies available in Weka can be used and compared by ACE, and that classifiers currently absent from Weka can be developed in the Weka framework and added to ACE as it matures.

Bayesian classifiers, nearest neighbour classifiers, neural networks, support vector machines and induction trees are just a few examples of the classifiers used by ACE. A variety of selection and fusion techniques are used to combine classifiers, including bagging and boosting approaches. Round-robin and hierarchical classification are explored in addition to flat classification where appropriate. In addition to experimenting with different classifiers, ACE also experiments with a variety of parameters for each classifier type.

4. FEATURES AND DIMENSIONALITY REDUCTION

ACE includes a variety of dimensionality reduction and feature weighting techniques. The evaluation of several techniques iteratively, such as binary feature selection followed by feature weighting, is also incorporated into ACE. Techniques used include principle component analysis, genetic algorithms, tree searches and forward-backward algorithms.

As discussed in Section 1, an important aspect of ACE is that users may use it easily with existing feature extraction systems. In order to facilitate communication with such systems, a simple but flexible XML file format has been developed for ACE. Alternatively, feature extractors may write their features to the commonly used Weka ARFF format, which is also understood by ACE.

Musical applications often require the analysis of multiple possibly overlapping frames belonging to a single piece of music. Furthermore, individual features may be multi-dimensional in a single frame. ACE and its XML file format have therefore been implemented so that they can easily deal with framed and/or non-framed feature vectors of arbitrary dimensions. Each features may be extracted for each frame or only once per recording. ACE also makes it possible to either classify frames individually or to classify a recording as a whole, as one may wish to base classifications on a methodology more complex than just a vote of individual frame classifications.

5. FEATURE EXTRACTION SUB-SYSTEMS

As stated above, ACE is designed to work with arbitrary existing feature extraction systems. However, default MIDI and audio feature extraction sub-systems are also included with ACE. This is to accommodate those who do not have easy access to feature selection systems and who wish to have the convenience of simply installing ACE and not worrying about implementing or installing any other software.

The MIDI sub-system is based on the feature extractor used by the Bodhidharma genre classification system [8]. This is the most extensive MIDI feature extractor available, to the best of the authors' knowledge.

The audio sub-system has been implemented in Java based on widely available algorithms included in systems such as MARSYAS [13]. This sub-system has been designed to minimize the effort required to add new fea-

tures and, in particular, to provide a simple mechanism for adding features that use previously extracted features in their calculations.

6. USING ACE

The first way to use ACE is through an easy-to-use and well-documented GUI, currently under development, that allows users to build taxonomies, label and manage training and testing instances, manage features, control classifier settings, carry out comparisons of classification methodologies, train and use classifiers and view results of experiments and classifications. This GUI is based in part on the Bodhidharma GUI [8].

The second way of using ACE is through a simple command-line interface. This interface is useful for users who already have the appropriate configuration files set up and would like a quick and easy method of performing tasks such as batch processing.

The final way of using ACE is for users to directly access the ACE Java classes from their own software. ACE is entirely open source, and it is distributed with good quality Javadocs and a manual describing the class structure and software implementation. This is useful for users who wish to integrate ACE's functionality directly into their own systems. ACE has been designed to be easily extensible, and users are free to add to it or change it.

The central purpose of ACE is to find a good classification methodology for any given problem. There can often be tradeoffs between different techniques in terms of success rates versus processing times. ACE makes it possible to meet the individual needs of particular users by allowing one to set constraints on training and/or testing times so that the best approach can be found within particular time limitations.

ACE includes four simple but flexible XML file formats that are used to store settings and to communicate with external software:

- **Taxonomy:** Specifies the classes that instances can belong to. Hierarchical structuring of classes is permitted. Allowances are made for modifications to the standard hierarchical structure, including the possibilities of having given instances belong to multiple classes or of allowing one class to be descended from multiple parent classes.
- **Feature Key:** Specifies the names and descriptions of the features that classifiers can use for a particular problem. This makes it easy to use arbitrary features extracted from arbitrary feature extraction systems.
- **Feature Vectors:** Specifies the particular feature vector values for sets of instances. This file format allows sequences of possibly overlapping feature vectors for frames of a particular instance as well as global feature vectors for each instance. More details are provided in Section 4.
- **Classifications:** Stores either model classifications or classification results. Functionality is included to allow different, possibly overlapping, segments of a single recording to belong to different classes. A given instance is permitted to belong to multiple classes simultaneously.

This file architecture deviates from the standard approach of using a single file to store both feature vectors and model classifications as well as, sometimes, the taxonomy of candidate classes. There are several reasons for this deviation.

Firstly, this enables a large set of features to be extracted once and then stored in a single file that can be used for multiple purposes. For example, features could be extracted from a large collection of recordings and stored in a single file that could be used to classify the recordings based on genre, composer and geographical point of origin, three tasks that require entirely different taxonomies.

Furthermore, one does not need to reprocess the model classifications if the features used change. For example, one could classify a given corpus of audio recordings and then later perform the same task on symbolic recordings of the same corpus using the same Classifications file.

The separation into multiple files makes it possible to distribute one type of XML file for arbitrary purposes to others without needing to impose one's own choices for the other three XML files. ACE also makes allowances for more general, flexible and sophisticated taxonomies and feature structures than most systems, and the separation into multiple files makes it easier to conceptualize sophisticated arrangements.

7. USING DISTRIBUTED COMPUTING

Many classification techniques are computationally intensive, especially when many features are used or there are large training sets. This is particularly problematic in a system involving multiple classifiers. Functionality is therefore currently being built into ACE to allow it to run on a number of computers in parallel. This will result in a reduction in runtime via known efficient and effective parallel implementations for tasks such as training and feature weighting/selection [14, 11]. Additionally, approaching some tasks in parallel, such as "coarse-grained" genetic algorithm optimizations, may affect the quality as well as the speed of the result [2].

Two Java-based distributed computing systems are currently being considered for use, namely Grid Weka [5] and M2K/D2K [3]. Grid Weka has the advantage of being built directly on the Weka toolkit. D2K is a well-established and powerful environment, but M2K itself is currently only in alpha release.

8. CURRENT APPLICATIONS

Several music-related classification problems are being used to test ACE, including genre classification, track segmentation of LPs, beat-box sound recognition and

snare drum timbre recognition. The data from the UCI Machine Learning Repository and Weka test sets are also being used to assess the capabilities of ACE.

9. CONCLUSIONS

ACE allows researchers to experimentally determine good classifiers, classifier combination methodologies and dimensionality reduction and feature weighting approaches for their specific classification needs, in terms of classification success rates, training times and classification times. ACE also provides music researchers with an easy-to-install and use, portable and extensible software package that supplies them with a powerful palette of classification techniques that they can use out-of-the-box. Users have the option of using either the bundled MIDI and audio feature extractors or their own customized feature extractor. Further information on ACE, including source code, will be made available on the ACE web site.

10. FUTURE RESEARCH

An expansion of ACE to allow the use of unsupervised and reinforcement learning is planned for the near future. Another priority is to include functionality for constructing sophisticated blackboard systems. A related goal is to make it possible to integrate knowledge sources based on heuristics into ACE.

Once the distributed aspect of the system is complete, a server-based sub-system will be designed that contains a coordination system and a database. Although not necessary for using ACE, users may choose to dedicate a computer to this server, allowing ACE to run perpetually. The server will keep a record of performances of all ACE operations run on a particular user's cluster and generate statistics for self-evaluation and improvement. The server, and possibly other computers in the cluster, will make use of any idle time to attempt to improve solutions to previously encountered but currently inactive problems.

Over time, gradual additions will be made to the ACE library of classifiers, dimensionality reducers and classifier combination techniques. The bundled feature extractors will also be expanded as time goes on.

It is hoped to eventually integrate ACE into the M2K framework.

11. ACKNOWLEDGEMENTS

The generous financial support of the *Social Sciences and Humanities Research Council of Canada* and the *Centre for Interdisciplinary Research in Music, Media and Technology(CIRMMT)* has helped to make this research possible.

12. REFERENCES

[1] Alpaydin, E. 2004. *Introduction to machine learning*. Cambridge, MA: MIT Press.

[2] Cantú-Paz, E. 2000. *Efficient and accurate parallel genetic algorithms*. Boston: Kluwer Academic Publishers.

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

[4] Duda, R., P. Hart, and D. Stork. 2001. *Pattern classification.* New York: Wiley.

[5] Khoussainov, R., X. Zuo, and N. Kushmerick. 2004. Grid-enabled Weka: A toolkit for machine learning on the grid. *ERCIM News* 59.

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

[7] Kuncheva, L. 2004. *Combining pattern classifiers: Methods and algorithms.* New York: Wiley.

[8] McKay, C. 2004. *Automatic genre classification of MIDI recordings.* M.A. Thesis. McGill University, Canada.

[9] Minaei-Bidgoli, B., G. Kortemeyer, and W. Punch. 2004. Optimizing classification ensembles via a genetic algorithm for a web-based educational system. *Proceedings of the International Workshop on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition*. 397–406.

[10] Pachet, F., and Z. Aymeric. 2004. Automatic extraction of music descriptors from acoustic signals. *Proceedings of the International Conference on Music Information Retrieval*. 353–6.

[11] Punch, W., E. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody. 1993. Further research on feature selection and classification using genetic algorithms. *Proceedings of the International Conference on Genetic Algorithms.* 557–64.

[12] Russell, S., and P. Norvig. 2002. *Artificial intelligence: A modern approach*. Upper Saddle River, NJ: Prentice Hall.

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

[14] Witten, I., and E. Frank. 2000. *Data mining: Practical machine learning tools and techniques with Java implementations.* San Francisco: Morgan Kaufmann Publishers.