COMBINING D2K AND JGAP FOR EFFICIENT FEATURE WEIGHTING FOR CLASSIFICATION TASKS IN MUSIC INFORMATION RETRIEVAL

Rebecca Fiebrink **Cory McKay** **Ichiro Fujinaga**
Music Technology
McGill University
Montreal, Canada
{rebecca.fiebrink, cory.mckay}@mail.mcgill.ca, ich@music.mcgill.ca

ABSTRACT

Music classification continues to be an important component of music information retrieval research. An underutilized tool for improving the performance of classifiers is feature weighting. A major reason for its unpopularity, despite its benefits, is the potentially infinite calculation time it requires to achieve optimal results. Genetic algorithms offer potentially sub-optimal but reasonable solutions at much reduced calculation time, yet they are still quite costly. We investigate the advantages of implementing genetic algorithms in a parallel computing environment to make feature weighting an affordable instrument for researchers in MIR.

Keywords: Classification, Feature Weighting, Parallel Computing, D2K.

1 INTRODUCTION

Classification is a lively area of research within music information retrieval (MIR). Genre and composer classification systems, similarity-based music recommendation systems, and intelligent interactive accompaniment systems are just a few of the areas where these techniques are used. Unfortunately, the variety and technical sophistication of pattern recognition techniques available can make it difficult to choose the best approach to apply to a particular problem.

An automated system for optimally selecting and fine-tuning classifiers for a given problem could allow researchers to devote their time and energies to tasks more important than adapting and implementing classification systems themselves. The Autonomous Classification Engine (ACE) project at McGill University is such a tool, and it is built with the particular needs of the MIR community foremost in mind (McKay et al. 2005). ACE experimentally compares a variety of dimensionality reduction techniques, classification algorithms, and classifier ensembles in order to find suitable approaches to use for a given user's data set, feature set, and taxonomy. The incorporation of efficient, parallel feature weighting using genetic algorithms will contribute significantly to ACE's power and flexibility; this paper discusses the implementation and performance of the feature weighting subsystem.

2 FEATURE WEIGHTING

Even though many features may be available to a classifier, it is not necessarily desirable to use all of them. In fact, the size of a classifier's training set must generally grow exponentially with the dimensionality of the feature space (Duda et al. 2001, 169–70). Relevant features can be selected from the set of available features by experimentation, wherein a classifier is trained and evaluated using candidate feature subsets. An exhaustive search for the optimal subset is often infeasible, however, because the number of potential subsets grows at a rate of $\Theta(2^d)$, where d is the number of available features (Siedlecki and Sklansky 1989).

Feature weighting attempts to further optimize a classification scheme by assigning real-valued weights to features according to their relevancy (Punch et al. 1993). Just as in feature selection, candidate sets of feature weights can be evaluated experimentally. However, the search space of candidate weight sets now grows with $\Theta(n^d)$, where n is the (potentially infinite) number of allowable values for each weight (Punch et al. 1993).

One might be tempted to perform an analysis of the final choice of feature subset or weights to obtain new insights about a classification problem, particularly in MIR. However, the performance of selection or weighting schemes is dependent on the data set, classification problem, and classifier in quite complex ways, and such analysis is beyond the scope of this project.

3 GA'S AND FEATURE WEIGHTING

Genetic algorithms (GA's) (Holland 1975) are an approach to computation inspired by biological evolution, and they are useful in optimization problems in which maxima are hard to find deterministically. A GA maintains a population of individuals that evolve according to specific rules of selection and through operators such as crossover and mutation. The fitness of each individual in the environment is evaluated, and selection exploits this information in favoring high-fitness individuals.

To accomplish feature selection or weighting for d features using GA's, individuals are represented as chromosomes with d genes, where the genes are binary-

valued for selection and real-valued for weighting. The fitness of each chromosome is evaluated experimentally by training and testing a classifier using the given weights, where better classifier performance yields higher fitness. Siedlecki and Sklansky (1989) found that GA's could quickly find near-optimal solutions to feature selection for a k-NN classifier. Punch et al. (1993) studied GA feature weighting for k-NN classifiers and found that feature weighting outperformed feature selection alone. They also found that implementing the system in parallel compensated for the long computation time needed to experimentally compute chromosome fitness. McKay (2004) has successfully applied GA's to feature selection and weighting in MIR classification problems. Additionally, Minaei-Bidgoli et al. (2004) have recently shown that GA's are also powerful tools for feature weighting in multiple classifier systems.

4 IMPLEMENTATION

This system includes functionality for feature weighting using GA's, and it allows for parallel configuration to reduce computation time. Of foremost concern to the design of any feature weighting system are its efficiency and accuracy. However, several other goals must be considered in the design of a system targeted at the diverse MIR community. The system should be portable to any operating system and hardware configuration. Its core features, including parallelism, should be accessible by any music research lab, whether it is outfitted with high-end shared-memory clusters or simply with a handful of workstations. The system should be scalable to use more or fewer computing nodes for its operations, depending on variable external demands on computing resources. Finally, the system should be flexible to allow customization (e.g., changes to the GA behavior).

4.1 JGAP

We looked to the open-source community for existing stable, robust GA software. JGAP (Rotstan and Meffert 2005) is a GA package written in Java and distributed freely. Its design is modular, it is well documented and easily extensible, and active developer and user communities promote continual improvements. These characteristics support our goals of portability and flexibility.

4.2 D2K

The Data-to-Knowledge (D2K) machine learning environment was chosen as the framework to support parallel and distributed operations. D2K is "a visual programming environment that allows for rapid prototyping and algorithm development" (Downie 2004). It is written in Java, so it is portable. Furthermore, it is the underlying foundation of the new M2K system for MIR research, so many of ACE's users will likely already be using D2K.

D2K programs (or "itineraries") are dynamically scalable in that a user can specify at runtime that individual components (or "modules") of a D2K program are to run in parallel, as well as designate on which remote machines they should run. There are no constraints on the hardware or operating systems of these machines, so D2K's parallelism is accessible to any lab with two or more computers connected over a LAN or the internet.

4.3 Implementation of the Feature Weighting System

Our feature weighting system combines a GA built on JGAP with custom D2K modules. The GA uses chromosomes of length d, where d is the number of potential features. Mutation occurs with a probability of 6.67%. Parents and children are all evaluated for fitness, and the top m individuals are preserved for the next generation, where m is the desired population size. (We keep the JGAP default settings for parameters such as mutation rate and selection method, as the goal is not to optimize GA weighting but rather to demonstrate its potential.) The algorithm stops when the best individual's fitness does not improve over five subsequent generations.

The parallel strategy used by the system is "master-slave parallelism" (Cantú-Paz 2000), also called "micro-grained parallelism" by Punch et al. (1993). In a master-slave GA, a single population is maintained on a "master" node that handles selection, crossover, and mutation. The master sends a portion of the population to each "slave" node for parallel fitness evaluation. The master-slave model is easy to implement, and existing design guidelines for serial single-population models can be directly applied (Cantú-Paz 2000).

The system uses a D2K itinerary to perform fitness evaluation on a population. An Input module loads the entire population and passes one chromosome at a time to a Fitness Evaluator module. The Fitness Evaluator assesses a chromosome's fitness via leave-one-out cross-validation of a k-NN classifier on the data. This module can clone itself and run in parallel on any number of slave machines, one clone at a time per processor. At its termination, each clone passes its chromosome with its calculated fitness to an Output module, which reassembles the population. Upon termination of the itinerary, control returns to the JGAP GA for selection.

This implementation allows slaves that are operating fastest at the time of execution to take on more of the computational load, as they can be assigned more module clones. This results in higher throughput than a system that naively places equal loads on all processors. This is of utmost importance to small labs, in which all nodes may double as workstations and/or web servers, and node performance can vary dramatically over the course of a day.

The current system is quite flexible: population size, GA behavior, the number of features, and the allowable feature weights can all be changed easily. Currently, the system evaluates chromosomes using a k-NN classifier derived from Weka and reads data from Weka ARFF files (Witten and Frank 2000), but a modular design allows use with any other classifier system.

5 SYSTEM PERFORMANCE

Three basic tests were conducted to assess system performance, and an additional two tests were run to further demonstrate its usefulness on problems for which exhaustive feature selection is infeasible. In the first test, the system performed feature weighting on a standard 768-instance, 8-feature data set (Pima Indian Diabetes from the UCI Repository, Blake and Merz 1998). Five trials were performed on each of three small populations, and k=1 neighbor was used. The master node used a 2.8GHz Pentium 4 PC with 1.5GB RAM, and the slaves used a 2.8GHz Pentium 4 server with 1GB RAM running Linux and a 867MHz, 512MB PowerPC running OS X.

Table 1 shows the population size, mean number of generations to convergence, mean hours to convergence, and the mean and standard deviation of the percent of instances correctly classified using leave-one-out cross-validation. Classification of this data set without weighting or selection yielded a success rate of 69.8%. An exhaustive search for the optimal feature subset took 0.27 hours and resulted in a success rate of 70.8%.

Table 1: Feature weighting, UCI Diabetes.

P	Avg. Gen.	Avg. Time (h)	Avg. % Correct	SD % Correct
10	8.8	0.109	72.5	0.475
20	10.2	0.230	72.1	0.687
50	18.6	0.944	73.2	0.590

In this small test, classification accuracy using feature weighting compared favorably to classification using optimal feature selection and classification without selection or weighting.

The second test examined classification time for another standard data set (UCI Breast Cancer, Zwitter and Soklic 1988), which contained 286 instances with 9 features. Its smaller size facilitated more extensive testing where ten tests were run on each population size: five tests used three nodes, and five used one (the master). The master ran on a 2.4GHz, 512MB RAM PC, and the slaves ran on the Linux and Macintosh machines used in Test 1. k=17 neighbors was used. Table 2 shows the population size, mean number of generations to convergence, mean and standard deviation of the percent correctly classified using leave-one-out cross-validation, and running time for one and three processors. Exhaustive search for the optimal feature subset took 0.08 hours and yielded a classification rate of 76.9%, and classification without selection or weighting yielded 73.4%.

Table 2: Feature weighting, UCI Breast Cancer.

P	Avg. Gen	Avg. % Correct	SD % Correct	Avg. Time (h)	
				1p	3p
10	10.8	76.9	0.66	0.04	0.03
20	11.4	77.2	0.43	0.10	0.05
50	10.2	77.5	0.47	0.27	0.08
100	11.2	77.8	0.54	0.58	0.16
200	13.4	78.1	0.46	0.84	0.35
500	11.5	78.2	0.51	2.09	0.59

Test 2 shows that feature weighting can be superior to exhaustive feature selection on this data set and compares favorably even for small populations. Test 2 also shows that classification accuracy tends to improve with population size, a finding supported by studies on GA's indicating that the greater diversity of large populations deters premature convergence and therefore tends to produce higher-quality solutions (Cantú-Paz 2000). This finding underscores the need for parallel systems that can reduce the long runtime of feature weighting using large populations.

Test 2 demonstrates a significant speedup resulting from parallel execution. The total runtime for three nodes is, on average, approximately one-third of that for one node alone. This suggests that communication costs are very low in comparison to the cost of fitness evaluation. These results suggest that additional nodes would appreciably further speed up computation, and that labs with only a few machines can still benefit from parallelism.

The third test applied feature selection to the snare drum timbre recognition problem presented by Tindale et al. (2004). The data set consisted of 1260 instances, created by three players playing each of seven snare drum strokes twenty times on three drums. Tindale's study included features from time-domain only and time- and frequency-domain measurements, extracted from the attack portion only and from 512-sample windows over the whole signal. Each of the four classification problems selected here uses a unique combination of these feature and signal types to distinguish among the seven strokes. Feature selection was performed once on each problem using an initial population size of 50. k=1 neighbor was used. Table 3 shows the classification problem, the number of available features, the performance of Tindale's best classifier on that problem, and the performance of our classifier after feature selection. 10-fold cross-validation is used for both our results and Tindale's. An exhaustive search for optimal feature selection was also performed on the time-domain features; this yielded 91.9% accuracy for the attack portion problem and 92.9% accuracy for the 512-sample problem, and it took 0.7 hours to perform each search.

Table 3: Comparison of snare drum timbre classification with Tindale et al. 2004.

Problem	No. Features	Tindale et al. %	Feature Selection %
All, attack	57	94.9	98.5
All, 512	35	93.0	95.5
Time, attack	8	90.8	91.9
Time, 512	8	90.9	92.9

Test 3 demonstrates improvement in all four problems over Tindale's best classifications of snare drum timbre. These improvements arose from just one run of a feature selection GA with a relatively small initial population. Additionally, the GA's found optimal solutions for the two 8-feature problems for which it was possible to exhaustively search the selection space. Based on the find-

ings above, it is likely that the selection on the other two problems was near-optimal, and that using feature weighting could result in even greater accuracy.

Finally, the system was run on two other classification problems for which exhaustive feature selection was infeasible. The first was the UCI vehicle recognition problem, in which 846 instances with 18 features were classified into 4 categories (Blake and Merz 1998). k=1 neighbor was used. Classification without selection yielded 69.5% accuracy. Using feature selection and an initial population of 50, the system achieved 75.5% accuracy.

The second classification task, another example from MIR, was a beat-box sound recognition problem using 1192 recorded instances, each belonging to one of five classes of hits (Sinyor et al. 2005). The data was collected from six beatboxers, and 24 potential features were extracted using spectral and temporal measurements. k=1 neighbor was used. Classification without selection yielded 93.3% accuracy. Using feature selection and an initial population of 50, the system was able to reach 94.7% accuracy.

6 CONCLUSIONS

We have implemented efficient and accurate feature selection and weighting using JGAP and D2K in the context of MIR and the ACE project. Tests show that k-NN classification using feature weighting outperforms unweighted classification and can surpass classification using exhaustive feature selection. Additionally, feature selection alone outperforms the best published results on snare drum timbre classification. Furthermore, the system's parallel implementation results in significant speedup using as few as three nodes. These results suggest that this system can be quite useful to MIR research, especially when applied to large classification problems for which exhaustive feature selection is infeasible.

Future work will include adding greater flexibility, such as the ability to automatically optimize GA parameters given constraints on time and computing resources. Plans are also in place to use the existing parallel framework to efficiently conduct empirical comparisons of GA's with other selection and weighting algorithms on a variety of classification problems. Further testing, particularly on MIR-related problems, will continue to elucidate the system's relative strengths and limitations.

ACKNOWLEDGEMENTS

We gratefully acknowledge support from the McGill University Max Stern Fellowship in Music, SSHRC, and the McGill Alma Mater Fund. We also thank Adam Tindale and Elliot Sinyor for sharing their data.

REFERENCES

Blake, C., and C. Merz. 1998. "UCI Repository of machine learning databases." <http://www.ics.uci.edu/~mlearn/MLRepository.html> University of California, Irvine, Department of Information and Computer Sciences. Accessed 13 April 2005.

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

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

Duda, R., P. Hart, and D. Stork. 2001. *Pattern classification*. New York: John Wiley & Sons, Inc.

Holland, J. H. 1975. *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

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

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.*

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.

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 5th International Conference on Genetic Algorithms*, 557–64.

Rotstan, N., and K. Meffert. 2005. JGAP: The Java genetic algorithms package. <http://jgap.sourceforge.net/> Accessed 13 April 2005.

Siedlecki, W., and J. Sklansky. 1989. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters* 10 (5): 335–47.

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.*

Tindale, A., A. Kapur, G. Tzanetakis, and I. Fujinaga. 2004. Retrieval of percussion gestures using timbre classification techniques. *Proceedings of the International Conference on Music Information Retrieval*.

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

Zwitter, M., and M. Soklic. 1988. This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.