Genetic Programming for Musical Sound Analysis

Róisín Loughran[1], Jacqueline Walker[1], Michael O'Neill[2], and James McDermott[2]

[1] University of Limerick, Limerick, Ireland
[2] NCRA, University College Dublin, Ireland
roisis@gmail.com

Abstract. This study uses Genetic Programming (GP) in developing a classifier to distinguish between five musical instruments. Using only simple arithmetic and boolean operators with 95 features as terminals, a program is developed that can classify 300 unseen samples with an accuracy of 94%. The experiment is then run again using only 14 of the most often chosen features. Limiting the features in this way raised the best classification to 94.3% and the average accuracy from 68.2% to 75.67%. This demonstrates that not only can GP be used to create a classifier but it can be used to determine the best features to choose for accurate musical instrument classification, giving an insight into timbre.

Keywords: Musical Information Retrieval, timbre, Genetic Programming.

1 Introduction

By definition [1] timbre is that quality of auditory sensation by which a listener can distinguish between two sounds of equal loudness, duration and pitch. Thus aurally distinguishing between musical instruments is largely based on the timbre of the instrument. Accurately describing timbre is therefore paramount to musical instrument identification. Unfortunately, unlike pitch and loudness, timbre has proven to be somewhat difficult to measure. To identify musical instruments, many studies have used timbral features to train and test classifiers developed using various Machine Learning techniques as discussed in Section 2. As yet Evolutionary Learning techniques have not been used extensively and Genetic Programming (GP) has not been tried at all. This study proposes to look at the use of GP for the evolution of an accurate musical instrument classifier.

Section 2 looks at previous developments in instrument identification over the past few decades and discusses the need for more specific feature selection. Section 3 describes the set up of the experiment. Section 3.1 and 3.2 introduce the instrument samples and timbral features used in the proposed study. Section 3.3 gives the details of the use of GP as an instrument classifier. Section 5 outlines the results obtained by the experiments, the significance of which are discussed in 6.

P. Machado, J. Romero, and A. Carballal (Eds.): EvoMUSART 2012, LNCS 7247, pp. 176–186, 2012.

2 Background

There have been a number of machine learning techniques used in creating musical instrument classifiers over the past two decades. Studies have used Mulitlayered perceptrons [11], k-Nearest Neighbour [3] and Support Vector Machines [5] among others. An exhaustive account of the methods used is given in [9]. Although each study reported strong results, objective comparison between the studies is difficult as they differ in respects other than in the classification methods chosen. Each individual study classifies a different number of instrument from a different number of samples using its own set of features. The number of features included varies widely from very few up to hundreds [6]. Likewise the number of instruments varies from 2 [2] to 30 [4]. Those that include a small number of samples across a wide range of instruments may not be general enough to identify notes played with varying pitch or dynamic. The proposed study avoids this problem by using a large number of samples varying in dynamic and model from just five instruments.

The trend with many of the previous studies mentioned above is to adopt the more is better theory; the inclusion of more and more detailed features from a given sound with a given classifier will automatically improve the accuracy of the classification. It was shown in [15] that this is not necessarily so and that the inclusion of extra features in a musical instrument classifier can actually *reduce* the accuracy of classification. Thus it is clear we need some intelligent way of deciding which features are beneficial and which are superfluous for accurate instrument identification. Genetic Algorithms (GA) have been used to some extent for this purpose with encouraging results. Unfortunately, GAs can only optimise a set of features — they do not classify them directly. GP on the other hand can directly evolve a classifier and in doing so determine which are the most and least important features to include, offering some insight into timbre.

3 Method

In any musical instrument classification experiment a number of issues have to be decided from the outset: what instruments to classify, what range of note pitches to examine, which features to incorporate and what classification method to use. This section examines these aspects of the current study.

3.1 Instrument Samples

This study incorporates a maximum of 95 features taken from 3006 samples of five instruments: the piano, violin, trumpet, flute and guitar. It was limited to five instruments to include maximum variation within each instrument, ensuring that the classifier is general enough to recognise any given sample from one of these instruments. The samples in this study were taken from the RWC Music Database (Musical Instrument Sound) [7] and the MUMS Database [17]. The RWC samples offer a number of models, dynamics and playing styles for each instrument. Three models of violin and guitar and two models of piano, flute and

trumpet were each sampled at dynamic levels f, mf and p across their entire pitch ranges. Where possible both vibrato and non-vibrato samples were included. The MUMS samples offered a further set of samples for the piano, violin, flute and trumpet. This results in 616 piano samples, 813 violin samples, 481 flute samples, 394 trumpet samples and 702 guitar samples. The total set of 3006 samples were split into 10 cross validation sets, 9 containing 300 samples and one containing 306 samples. For this study, 9 sets containing 2706 note samples were used to train the programs and the remaining 300 'unseen' notes were used as validation samples.

3.2 Timbral Features

Previous studies discussed in Section 2 used a wide variety of timbral features calculated in a number of ways. The current study hopes to choose the best features to use but in doing so must also start with a large number of features, some of which may turn out to be redundant. The features included were chosen from those used in the previous studies. The list of temporal and spectral features used initially is given in Table 1. Details of the calculation of these features may be found in [13].

Table 1. List of temporal and spectral features included initially in this experiment

Temporal	**Spectral**
Temporal Envelope	Spectral Envelope
Residual Envelope	Number Spectral Peaks
Attack Time	Irregularity
Attack Slope	MFCCs (1-16)
Centroid Envelope	Inharmonicity
Zero-Cross Rate	Centroid
Number Onsets	Spread
Onset Distance	Skew
	Kurtosis
	Regularity
	Rolloff
	Brightness

A number of the features listed in Table 1 contain multiple data points per feature. As much of this data is redundant, it was reduced using Principal Component Analysis (PCA) [10]. PCA transforms the data orthonormally, maintaining the variance of the data but concentrating it into the lower dimensions. This results in a set of principal components, with variance ordered from highest to lowest. PCA was implemented in Matlab [16] using the princomp function from the Statistics Toolbox. The first four principal component values for the temporal, residual and spectral envelopes, evolution of the centroid and the temporal envelope of each of the 16 MFCCs were included. This resulted in 95 data points across all features.

3.3 Genetic Programming

GP, like GA, is based on the concept of *natural selection.* GP starts with an initial population of random individual solutions to a given problem and mutates and combines them over a number of generations to improve the performance of the population. This continues until either a given performance or *fitness* is reached or a specific number of generations have passed. The difference between GP and GA is that whereas GA evolves binary or floating point strings, GP evolves solutions that consist of executable programs. These programs are represented by structures such as the tree structures used in this study. Each program tree is created from a user-defined set of *terminals* and *functions.* The internal nodes of the trees consist of functions whereas each external leaf of the tree consists of one of the allowed terminals. The GP in this study was implemented using the GPLAB Toolbox for Matlab [18]. The terminals of each program tree comprised of the 95 features listed in Table 1 and a number of assigned constants. GPLAB offers a large number of possible arithmetic and logical function to the user, many of which may be over-complicated for the purpose of this study. Thus the functions allowed were limited to very simple mathematical and logical functions as detailed in the Section 4.

Fitness Function. The *fitness* is measured as how often each individual correctly recognises a musical instrument. Each individual returns a numerical value to indicate which instrument it has identified. This fitness determines whether or not it will survive to the next generation of solutions. Two different fitness values *single* and *range* were used in this experiment as detailed in the following section.

4 Experiment

As discussed above, the full 95 by 3006 data set was split into 10 cross validation sets. For these experiments 9 of these sets (containing 2706 samples) were used to train the GP and the remaining 300 'unseen' samples were used to validate. A random population of 100 individual program trees was created. These trees were created using the ramped half-and-half method to ensure a diverse population. The fitness of each individual indicated how many note samples it mis-classified. A minimising fitness function was used, thus the optimal fitness was 0 where no samples were misclassified and the worst was 2706 whereby all training samples were misclassified. Initially, using the *single* fitness method, the target was set so that 1 corresponded to piano, 2 to violin, 3 to flute, 4 to trumpet and 5 to guitar. To ensure these numbers can be calculated by the individual programs, the terminals allowed by each program consisted of the feature values and the constants 1, 2, 3, 4, 5 and any floating point value between 0 and 1. These targets are quite specific to match however and judgement must always be shown in developing any fitness function [12]. It is possible that it may be easier for the GP to evaluate values to a range of targets rather than an individual value and so the *range* fitness was employed which transformed the targets to a wider range of values. Thus the fitness targets became:

- (1 - 10) = Piano
- (11 - 20) = Violin
- (21 - 30) = Flute
- (31 - 40) = Trumpet
- (41 - 50) = Guitar

Most of the calculated feature values are between 0 and 1 making it unlikely that the classifiers will often reach targets up to the value of 50. Thus to ensure that the individual programs can generate values that high, further constants — 5, 10, 20, 30 and 40 were included as allowed terminals in these experimental runs. The fitness of each individual was calculated as the sum of the difference between the target vector and the output of the individual rounded to an integer.

As mentioned above, GPLAB offers a wide variety of functions in creating the individual solutions. We wished to keep the evolved programs simple, focussing on a pure combination of the extracted features. Thus the allowed functions were initially limited to the arithmetic function *plus*, *minus* and *times*. However the relationship between certain features may also be of interest. A strong output should *Residual(2)* be greater then *Centroid(1)* (an 'unexpected' relationship), for example, could be useful for determining an unusual relationship between features — thus possibly identifying a particularly difficult timbre. Hence, another experiment was run including the boolean operators *less than* and *greater than* along with the arithmetic operators.

The combination of different fitness measures and functions were combined so that four experiments were run and compared:

- Single Math — single value fitness method using just arithmetic operators
- Single All — single value fitness method using arithmetic and boolean operators *greater than* and *less than*
- Range Math — range fitness method using just arithmetic operators
- Range All — range fitness method using arithmetic and boolean operators *greater than* and *less than*

Bloat is a problem that can occur with GP whereby the program tree continues to grow in size without any significant improvement in performance. It is an issue as it can lead to unnecessarily large programs but can be prevented by limiting the allowed size or depth of the programs trees. To prevent bloat the *dynamic* level was set to 6 and the *realmaxlevel* set to 15 using GPLAB. These values are set so that if a tree is created that is greater in size than the dynamic level but lower than realmaxlevel it is checked against the current best individual. If it is better than the current best individual the dynamic level is increased and the individual is passed to the next generation. Also to favour smaller individuals the sampling method used was *lexictour*. This is similar to the tournament method of sampling in that parents are chosen by randomly drawing a number of individuals from the population and selecting from the best group. The difference with lexictour is that if two individuals are equally fit, the shorter individual will be favoured. The number of individuals in each population was 100 and each experiment was run for 500 generations. These 4 experiments were repeated 30 times and the best individual from the population at the end of each run was noted.

5 Results

Each of the program trees was evolved using the features from 2706 note samples. To verify the classification accuracy of the evolved programs, each was used to classify the remaining 300 unseen note samples. The 30 best program trees evolved by each of the four methods were used to evaluate each of the 300 notes. The classification results were given as the percentage of times each program correctly identified an instrument from the given note samples. A boxplot of the results obtained by the four methods is shown in Figure 1. This shows the dispersion of accuracy of the best 30 evolved programs for each of the 4 methods. It can be seen from this plot that the highest mean accuracy (across the 30 runs) is from the Single All method at 64%. This method also produced the solution with highest overall classification results of 88.3%. To attempt to improve this result further, the most successful method Single All was run again for a longer amount of time.

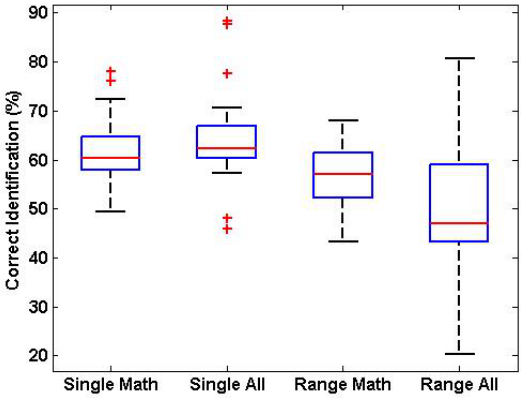

Fig. 1. Classification results for the 300 element test set by each of the four fitness methods

5.1 Extended Run

The experiment with single value fitnesses and both arithmetic and boolean operators (the Single All method) was run 30 times with a population of 100 for 2000 generations. Larger trees were encouraged to evolve in this run with the dynamic level set to 15 and the realmaxlevel set to 25. The evolved programs were again tested by using them to classify the unseen set of 300 samples. A boxplot of the results obtained is shown in Figure 2. Again this shows a wide variation in the classification accuracy of the best evolved programs but the highest classifications have improved. More details of the five most successful programs is given in Table 2. This shows the size and training and test accuracy of the best individuals. It can be seen that three individual programs achieved classification accuray of over 90% on unseen samples with the highest achieving 94%.

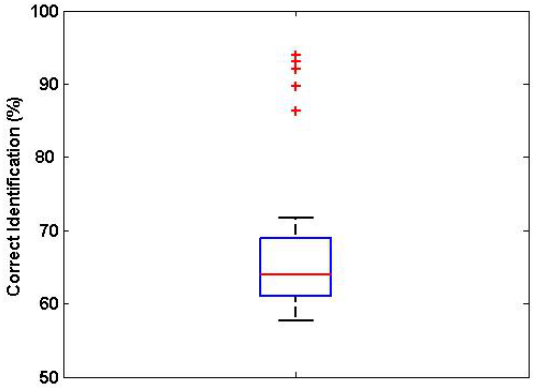

Fig. 2. Classification results for the 300 element test set by the Single All extended run

Table 2. The depth, number of nodes, training fitness and classification accuracy for each of the 5 best evolved strings

Ind.	Depth	#Nodes	Train(/2706)	Train(%)	Test Class(%)
9	23	241	174	93.6	94
2	19	245	149	94.5	93
15	20	173	155	94.3	92
30	23	225	193	92.8	89.67
11	14	67	316	88.3	86.33

As the results were so vastly improved by increasing the number of generations, it was thought that further improvements may be found by extending the size of the population. Thus the experiment was run again with a population of 500 over 100 generations to see if having a larger population would improve the average individual being passed on to the next generation. This was not found to improve the results however. The best individual was found to have a test classification accuracy of 87.3% dropping to 76% for the second best.

5.2 Limiting GP to 'Popular' Features

One of the best aspects of GP is that it allows the user to analyse the evolved programs to determine some of the more important aspects of sound. The individuals created using GP do not use all 95 features due to limitations on their size. Thus along with deciding how to combine the features used, GP must also decide which features to include. The simplest way to analyse the programs is to total the amount of times each feature was used by the more successful programs. The top 14 most popular features chosen by the GP runs is shown in Table 3. In an attempt to reduce the variation within the individuals, GP was run again with just these more popular features.

Table 3. The top 14 features as found from the GP Experiments

Feature No.	#Instances	Feature
2	22	Rolloff
9	23	MIRCent
10	20	MIRSpread
16	57	Env1
17	27	Env2
18	35	Env3
20	80	Cent1
21	40	Cent2
28	41	Spec1
33	29	MFCC1-2
37	23	MFCC2-2
40	20	MFCC3-1
42	22	MFCC3-3
44	51	MFCC4-1

The results of the best 5 individuals evolved using only the top 14 features is shown in Table 4. The best program evolved was from individual 9 which had a training fitness of 94.4% and a test classification accuracy of 94.3% — a slight improvement on the best classification accuracy using all features. Overall the average test classification rose from 68.2% to 75.67% across all 30 individuals using a limited number of features with 6 individuals achieving a test classification accuracy above 89%. This again demonstrates that accurate instrument identification is dependent on a careful selection of features, rather than merely incorporating as many features as one can in a given classifier.

From the depths of the programs in Table 2 and Table 4 it is clear that limiting the amount of features did not lead to smaller programs as may have been expected. By reducing the number of available features it was thought that the program might not grow as much. In contrast to this, the reduction in features coincided with an increase in average depth from 18 to 22 and an average number of node from 103 to 188. This may indicate that by only allowing 'good' features, extra branches to the trees (or extra code to the programs) give better classification and hence are kept.

Table 4. The depth, number of nodes, training fitness and classification accuracy for each of the 5 best evolved individuals using only the top 14 features

Ind.	Depth	#Nodes	Train(/2706)	Train(%)	Test Class(%)
9	21	263	151	94.4	94.3
23	22	293	174	93.6	93.7
2	17	151	170	93.7	93.3
13	17	115	266	90.2	90.3
7	20	161	246	90.9	89.3

5.3 Comparison with Other Methods

This work was carried out as part of a larger study detailed in [13] which examined a number of methods for musical instrument classification. An MLP was used as a classifier along with a reduced feature MLP with features selected using a floating point GA [14]. Each of these methods were tested and compared with three sets of verification samples. The *General* set comprised of the 300 unseen samples as used in the above study, the *One-octave* set comprised of one octave of notes C4-C5 common to each instrument and the *Listening* set comprised of a selected group of notes that due to pitch or volume were considered particularly difficult to aurally recognise. A comparison of the results is shown in Table 5.

Table 5. Comparison of the GP classifiers with MLP and GA-MLP classifiers tested with one-octave samples, general samples and the listening samples

Set	MLP	GA-MLP	GP	GP Reduced
1 Octave	70.31%	50.02%	53.85%	70.77%
General	99.63%	99.3%	94%	94.3%
Listening	45.21%	44.34%	46.6%	56.2%

As can be seen from this table, the reduced GP outperforms the MLP and the GA-MLP for both the One-Octave and Listening test samples. Although it does not perform as well on the General set of samples it must be noted that the evolved programs are considerably less complicated and less computationally expensive to run than the MLP used in the other two methods. Using GP to create a musical classifier also has the advantage that the resulting programs are accessible and easily analysed by the user. The Listening set of samples proved particularly difficult for each classifier. As a benchmark for the classifiers, this set of samples was given as a listening test to thirty human volunteers. Although the results of these tests varied from 57% to 97%, the average result was 83% — considerably higher than any of the developed classifiers. This indicates that even a statistically successful classifier does not outperform the human ear. It also demonstrates the importance of the selection of samples when verifying a classifier. Many of the studies described in Section 2 do not give specific details of the samples chosen for verification yet the results above indicate that this is crucial to the accuracy of the representation of your results. Ideally a set of high quality audio samples would be publicly available and standardised for this purpose. Further details of each of these experiments may be found in [13].

6 Conclusion

This study explored the use of GP in developing musical instrument classifiers. Programs were evolved that achieved over 94% classification accuracy on unseen samples. These results compared favourably with similar experiments using

MLPs and GAs as detailed in [13]. As noted in Section 3 these samples covered the full natural pitch range of each instrument played at various levels of loudness. Although the programs took a long time to evolve — over 10 days for some of the longer runs — the resultant programs are very simple. These evolved programs based on a simple combination of a number of extracted timbral features are much simpler than some of those classification methods discussed in Section 2. In addition to this, the evolved programs are accessible to the user. This means that we can analyse the best programs to determine which features are the most important to use. In doing this we reduced the number of features from 95 to 14 while increasing the best classification accuracy from 94% to 94.3% and the average classification accuracy among the best 30 programs from 68.2% to 75.67%. The selection of features originally included in this study were chosen from those included in similar classification experiments. The fact that we achieved higher classification with fewer features indicates that features should be more carefully selected for such experiments. It is possible that the inclusion of a large number of features may offer too much or conflicting information to a given classifier, causing a reduction in classification accuracy. This is a clear indication that in the case of choosing features for musical instrument identification less is *not* more; the accuracy of any classifier is dependent on the selection of features it classifies with.

To continue this work, we hope to further analyse the features most often selected by the GP to determine if there is some reason as to why these make better timbral descriptors than other features. Timbre description studies such as those in Multi-dimensional Scaling [8] try to reduce the number of descriptors for timbre and yet timbre classification studies such as those described in Section 2 seem to incorporate more and more features into their classifiers. We hope that in using Evolutionary Methods — in particular GP — we may be able to bridge this gap by determining which are the most important timbre features for accurately describing the timbre of musical instruments. Further to this we would like to consider evolving programs on the raw musical data to see if it can evolve its own features. If any successful programs are evolved we could then analyse the structure of these programs to possibly determine a new set of features for timbre description.

Acknowledgments. This study is funded by the Science Federation of Ireland (SFI), under the current National Development Plan and Strategy for Science Technology and Innovation (SSTI) 2006-2013.

References

1. ANSI: American National Standard-Psychoacoustical Terminology. American National Standards Institute, New York (1973)
2. Brown, J.: Computer identification of musical instruments using pattern recognition with cepstra coefficients as features. Journal of Acoustical Society of America 105(3) (1999)

3. Eronen, A.: Comparison of features for musical instrument recognition. In: Workshop on Application of Signal Processing to Audio and Acoustics, New York, pp. 19–22 (2001)
4. Eronen, A., Klapuri, A.: Musical instrument recognition using cepstral coefficients and temporal features. In: ICASSP (2000)
5. Essid, S., Richard, G., David, B.: Musical instrument recognition by pairwise classification strategies. IEEE Transaction on Audio, Speech and Language Processing 14, 1401–1412 (2006)
6. Fraser, A., Fujinaga, I.: Toward real-time recognition of acoustic musical instruments. In: ICMC, China, pp. 175–177 (1999)
7. Goto, M.: Development of the RWC music database. In: Proceedings of the 18th Congress on Acoustics (2004)
8. Grey, J.: Multidimensional perceptual scaling of musical timbres. Journal of Acoustical Society of America 61, 1270–1277 (1977)
9. Herrera, P., Amatriain, X., Batlle, E., Serra, X.: Towards instrument segmentation for music content description: A critical review of instrument classification techniques. In: ISMIR, Plymouth, Massachusetts (2000)
10. Hotelling, H.: Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology 24, 417–441, 498–520 (1933)
11. Kaminskyj, I., Materka, A.: Automatic source identification of monophonc musical instrument sounds. In: IEEE International Conference on Neural Networks, pp. 189–194 (1995)
12. Koza, J.: Genetic evolution and co-evolution of game strategies. In: The International Conference on Game Theory and Its Applications. Stony Brook, New York (1992)
13. Loughran, R.: Musical Instrument Identification with Feature Selection Using Evolutionary Methods. Ph.D. thesis, University of Limerick (2009)
14. Loughran, R., Walker, J., O'Neill, M.: An exploration of genetic algorithms for efficient musical instrument identification. In: Irish Signals and Sytems Conference, Dublin, Ireland (2009)
15. Loughran, R., Walker, J., O'Neill, M., O'Farrell, M.: Comparison of features for musical instrument identification using artificial neural networks. In: CMMR, Copenhagen, Denmark, pp. 19–33 (2008)
16. MATLAB7: Version 7.2 (r2006a) matlab software. In: The Mathworks (2006)
17. Opolko, F., Wapnick, J.: McGill university master samples (MUMS) (cds) (1987)
18. Silva, S.: GPLAB - a genetic programming toolbox for MATLAB (2004), `http://gplab.sourceforge.net/`