Musical Source Separation using Generalised Non-Negative Tensor Factorisation models

Derry FitzGerald DERRY.FITZGERALD@CIT.IE
Matt Cranitch MATT.CRANITCH@CIT.IE
Dept. of Electronic Engineering, Cork Institute of Technology, Rossa Avenue, Cork, Ireland

Eugene Coyle EUGENE.COYLE@DIT.IE
School of Electrical Engineering Systems, Dublin Institute of Technology, Kevin Street, Dublin, Ireland

Keywords: Musical Sound Source Separation, Tensor Factorisation

Abstract

A shift-invariant non-negative tensor factorisation algorithm for musical source separation is proposed which generalises previous work by allowing each source to have its own parameters rather a fixed set of parameters for all sources. This allows independent control of the number of allowable notes, number of harmonics and shifts in time for each source. This increased flexibility allows the incorporation of further information about the sources and results in improved separation and resynthesis of the separated sources.

1. Introduction

In recent years, machine learning techniques such as shift-invariant non-negative matrix and tensor factorisation have received much attention as a means of separating sound sources from single and multichannel mixtures (Mørup M., 2006; Virtanen, 2006). Using shift-invariance in frequency allows a pitched musical instrument to be modelled as a single frequency basis function which is translated up and down in frequency to model different notes played by an instrument, while shift-invariance in time allows the temporal evolution of a sources timbre to be captured.

More recently, improvements have been made to these models by imposing harmonicity constraints on pitched instruments (FitzGerald et al., 2008). This was done by using an additive synthesis model where a pitched instrument is modelled by a set of harmonic weights. A source-filter model was also incorporated to allow the timbre of instruments to change with pitch, resulting in improved separations. Further, the use of harmonicity constraints restricts the decompositions sufficiently to allow simultaneous separation of pitched and unpitched instruments. However, for best results, the model now requires an estimate of the pitch of the lowest note played by each pitched instrument.

However, a limitation of shift-invariant models to date is that all the sources have to have the same parameters. For example, the number of allowable notes or shifts in frequency is the same for all pitched instruments. However, the range of notes played in a given piece will vary with the instrument. It can be seen that being able to set the number of notes for each instrument individually will reduce the possibilities for error in the separation. This would be of particular use in score-assisted separation, such as proposed in (Woodruff et al., 2006). Further, the number of harmonic weights required to model a given source varies. For example, modelling a flute will typically require less harmonics than a piano or violin. Therefore, the ability to vary the parameters for each source individually would be advantageous, and would help to improve the separations obtainable using shift-invariant factorisation models.

2. Generalised Factorisation Models

In the following, $\langle \mathcal{AB} \rangle_{\{a,b\}}$ denotes contracted tensor multiplication of $\mathcal{A}$ and $\mathcal{B}$ along the dimensions a and b of $\mathcal{A}$ and $\mathcal{B}$ respectively. Outer product multiplication is denoted by $\circ$. Indexing of elements within a tensor is notated by $\mathcal{A}(i, j)$ as opposed to using subscripts. This notation follows the conventions used in the Tensor Toolbox for Matlab, which was used to implement the following algorithm (Bader & Kolda, 2007). For ease of notation, as all tensors are now instrument-specific, the subscripts are implicit in all tensors within summations.

Given an r-channel mixture, spectrograms are obtained for each channel, resulting in $\mathcal{X}$, an $r \times n \times m$ tensor where n is the number of frequency bins and m is the number of time frames. The tensor is then modelled as:

$$\mathcal{X} \approx \hat{\mathcal{X}} = \sum_{k=1}^{K} \mathcal{G} \circ \langle \langle \mathcal{R}\mathcal{W} \rangle_{\{3,1\}} \langle \mathcal{S}\mathcal{P} \rangle_{\{2,1\}} \rangle_{\{2:3,1:2\}} + \sum_{l=1}^{L} \mathcal{M} \circ \langle \mathcal{B} \langle \mathcal{C}\mathcal{Q} \rangle_{\{1,1\}} \rangle_{\{2,1\}} \qquad (1)$$

with $\mathcal{R} = \langle \mathcal{F}\mathcal{H} \rangle_{\{2,1\}}$ and where the first right-hand side term models pitched instruments, and the second unpitched or percussion instruments. K denotes the number of pitched instruments and L denotes the number of unpitched instruments.

$\mathcal{G}$ is a tensor of size r, containing the gains of a given pitched instrument in each channel. $\mathcal{F}$ is of size $n \times n$, where the diagonal elements contain a filter which attempts to model the formant structure of an instrument, thus allowing the timbre of the instrument to alter with frequency. $\mathcal{H}$ is a tensor of size $n \times z_k \times h_k$ where z_k and h_k are respectively the number of allowable notes and the number of harmonics used to model the kth instrument, and where $\mathcal{H}(:, i, j)$ contains the frequency spectrum of a sinusoid with frequency equal to the jth harmonic of the ith note. $\mathcal{W}$ is a tensor of size $h_k \times p_k$ containing the harmonic weights for each of the p_k shifts in time that describe the kth instrument. $\mathcal{S}$ is a tensor of size $z_k \times m$ which contains the activations of the z_k notes associated with the kth source, and in effect contains a transcription of the notes played by the instrument. $\mathcal{P}$ is a translation tensor of size $m \times p_k \times m$, which translates the activations in $\mathcal{S}$ across time, thereby allowing the model to capture temporal evolution of the harmonic weights.

For unpitched instruments, $\mathcal{M}$ is a tensor of size r containing the gains of an unpitched instrument in each channel, $\mathcal{B}$ is of size $n \times q_l$ and contains a set of frequency basis functions which model the evolution of the timbre of the unpitched instrument with time where q_l is the number of translations in time used to model the lth instrument. $\mathcal{C}$ is a tensor of size m which contains the activations of the lth instrument, and $\mathcal{Q}$ is a translation tensor of size $m \times q_l \times m$ used to translate the activations in $\mathcal{C}$ in time.

A suitable metric for measuring reconstruction of the original data is the generalised Kullback-Leibler divergence proposed for use in non-negative matrix factorisation by (Lee & Seung, 1999):

$$D\left(\mathcal{X} \parallel \hat{\mathcal{X}}\right) = \sum \mathcal{X} \log \frac{\mathcal{X}}{\hat{\mathcal{X}}} - \mathcal{X} + \hat{\mathcal{X}} \qquad (2)$$

Using this measure, iterative update equations can be derived for each of the model variables.

The model has been used to separate stereo mixtures containing both pitched and unpitched instruments, and the ability to set the parameters independently has been observed to improve the separation results obtained in comparison to the situation where the same parameters are used for all instruments.

3. Conclusions

An algorithm was proposed which generalises previous work in shift-invariant non-negative tensor factorisation algorithms by allowing each source to have its own parameters, such as note range, number of harmonics and time shifts. This increased flexibilty has been observed to result in improved musical source separation performance for mixtures of pitched and unpitched instruments.

Acknowledgements

This research was supported by Enterprise Ireland.

References

Bader, B., & Kolda, T. (2007). Matlab tensor toolbox version 2.2.

FitzGerald, D., Cranich, M., & Coyle, E. (2008). Extended non-negative tensor factorisation models for musical sound source separation. *to appear in Computational Intelligence and Neuroscience.*

Lee, D., & Seung, H. (1999). Learning the parts of objects by non-negative matrix factorisation. *Nature*, *401*, 788–791.

Mørup M., Schmidt, M. (2006). *Sparse non-negative tensor 2d deconvolution (sntf2d) for multi channel time-frequency analysis* (Technical Report). Technical University of Denmark.

Virtanen, T. (2006). *Sound source separation in monaural music signals.* Doctoral dissertation, Tampere University of Technology.

Woodruff, J., Pardo, B., & Dannenberg, R. (2006). Remixing stereo music with score-informed source separation. *Proceedings of the 7th International Conference on Music Information Retrieval.*

Learning Violinist's Expressive Trends

Miguel Molina-Solana MIGUELMOLINA@UGR.ES
Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

Josep Lluis Arcos ARCOS@IIIA.CSIC.ES
Artificial Intelligence Research Institute, IIIA, CSIC, Campus UAB, 08193 Barcelona, Spain

Emilia Gomez EGOMEZ@IUA.UPF.EDU
Music Technology Group, Pompeu Fabra University, 08003 Barcelona, Spain

Abstract

This paper presents a Trend-based model for identifying professional performers in commercial recordings. Trend-based models characterize performers by learning how melodic patterns are played. Reported results using 23 violinists show the high identification rates achieved with our model.

1. Introduction

Expressive performance analysis and representation is a key challenge in the sound and music computing area. Previous research has addressed expressive music performance using machine learning techniques. To cite some, in (Saunders et al., 2008) they represent pianists' performances as strings; in (Ramirez et al., 2007) they study how to measure performance aspects applying machine learning techniques; and in (Stamatatos & Widmer, 2005) a set of simple features for representing stylistic characteristics of piano music performers is proposed.

We focus on the task of identifying professional violinists from their commercial audio recordings. Our approach is based on the learning of the *Trend Models* that characterize each performer. Trend models capture expressive tendencies and are learnt from audio descriptors obtained by using state-of-the-art audio feature extraction tools.

The whole process is done in an automatic way, using only the recordings (scores are not used). Since commercial recordings are so heterogeneous, it is really difficult to exactly translate the audio to an accurate score representation. We deal with this problem by using a more abstract representation that the real notes, but still close to the melody (i.e. instead of focusing on the absolute notes, we focus on the melodic surface). Specifically, we deal with this task by (1) using a higher-level abstraction of the automatic transcription focusing on the melodic contour (Grachten et al., 2005); (2) tagging melodic segments according to the Implication-Realization (IR) model (Narmour, 1992); and (3) characterizing the way melodic patterns are played as probabilistic distributions.

2. Trend-Based Modeling

A trend model characterizes, for a specific audio descriptor, the relationships a given performer is establishing among groups of neighbor musical events. For instance, the trend model for the energy descriptor will relate, qualitatively, the changes of energy for a given set of consecutive ascending notes. The main processes of the system are:

Feature Extraction and Segmentation The first process consists on extracting audio features from recordings using an existing tool (Camacho, 2007). Using fundamental frequency information, note boundaries are identified and melodic segmentation is performed. For each note we collect its pitch, duration and energy. We used the IR model by E. Narmour to perform melodic segmentation. Each segment is tagged with its IR pattern.

Learning Trend Models Trend models capture the way different audio descriptors change in the different IR patterns. A trend model is represented by a set of discrete probability distributions for a given audio descriptor.

To generate trend models for a particular performer and note descriptor, we use the sequences of values extracted from the notes identified in each segment. From these sequences, each value is compared with respect to the mean value of the fragment and is transformed into two qualitative values meaning 'the descriptor value is higher than the mean', and 'the value is lower than the mean'. In the current approach, since we are segmenting the melodies in groups of three notes and using 2 qualitative values, eight (2^3) different patterns may arise.

Next, a probability distribution per IR structure with these patterns is constructed by calculating the percentage of occurrence of each pattern. Thus, trend models capture statistical information of how a certain performer tends to play. Combining trend models from different audio descriptors, we are improving the characterization of each performer. Trend models for both duration and energy descriptors have been learnt.

Classifying new recordings We are using a nearest neighbor (NN) classifier to generate a ranked list of possible performers for a new input recording. When a new recording is presented to the system, the feature extraction process is performed and its trend model is created. This trend model is compared with trend models learnt in the previous stage acting as class patterns.

The distance between two trend models, is defined as a weighted sum of the distances between their respective IR patterns (i.e. their respective probability distributions).

3. Results

We work with Sonatas and Partitas for solo violin from J.S. Bach. We tested our system by performing experiments with commercial recordings from 23 different violinists and using three movements: Mov.2 of Partita 1, Mov.6 of Partita 1, and Mov.5 of Partita 3. Each experiment consisted in learning trend models with one movement and then testing them with another movement. Figure 1 reports the results achieved in the experiments. Mov.2 versus Mov.6 experiments demonstrate the performance of the system by using two movements from the same piece. Mov.6 versus Mov.5 shows the performance of the system by using two movements from different pieces.

In experiments using movements from the same piece, the correct performer was majority identified in the first half of the list, while in movements from different pieces, the most difficult scenario, the 90% of identification accuracy is overcame at position 15. We can also observe that a 50% of success is achieved using the five first candidates in any case (doubling the 22% of a random classifier).

These results show that the model is capable of learning performance patterns that are useful for distinguishing performers. The results are promising, espe-

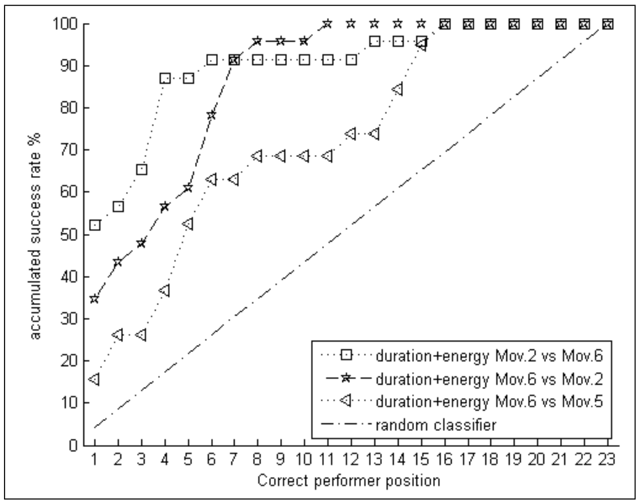

Figure 1. Accumulated success rate

cially comparing with a random classification where the success rate is clearly outperformed.

We plan to increase the number of descriptors and to experiment with the rest of movements.

References

Camacho, A. (2007). *Swipe: A sawtooth waveform inspired pitch estimator for speech and music.* Doctoral dissertation, University of Florida, USA.

Grachten, M., Arcos, J. L., & Lopez de Mantaras, R. (2005). Melody retrieval using the implication/realization model. *MIREX 2005.*

Narmour, E. (1992). *The analysis and cognition of melodic complexity: The implication realization model.* Chicago, IL: Univ. Chicago Press.

Ramirez, R., Maestre, E., Pertusa, A., Gomez, E., & Serra, X. (2007). Performance-based interpreter identification in saxophone audio recordings. *IEEE Trans. on Circuits and Systems for Video Technology*, *17*, 356–364.

Saunders, C., Hardoon, D., Shawe-Taylor, J., & Widmer, G. (2008). Using string kernels to identify famous performers from their playing style. *Intelligent Data Analysis*, *12*.

Stamatatos, E., & Widmer, G. (2005). Automatic identification of music performers with learning ensembles. *Artif. Intell.*, *165*, 37–56.

The Potential of Reinforcement Learning for Live Musical Agents

Keywords: interactive musical agents, reinforcement learning, computer music

Nick Collins N.Collins@sussex.ac.uk
Department of Informatics, University of Sussex, Falmer, Brighton, BN1 9QJ, UK

Abstract

Reinforcement learning has great potential applicability in computer music, particularly for interactive scenarios and the production of effective control policies for systems. This paper considers in particular the case of interactive music systems, where a software agent can be trained online during a rehearsal session with a musician. A small-scale system is described which uses the Sarsa(λ) algorithm to update state-action values to determine a policy for output behaviours over a modest feature space. It is hoped that issues arising can be discussed fruitfully in a workshop context.

1. Introduction

One holy grail of computer music is the creation of a live musical agent whose behaviours are fully commensurate with human music making, but whose architecture is not of flesh and blood (Rowe, 2001; Collins, 2007). Whilst many projects have been undertaken, most have sought out the stimulating but inhuman properties of certain algorithms for their own sake, or have otherwise failed to live up to stricter definitions of autonomous agency (Collins, 2006). The most promising research to date is probably that which utilises machine learning techniques (Thom, 2003; Hamanaka et al., 2003; Assayag et al., 2006). This paper considers the potential in particular of reinforcement learning (RL) (Sutton & Barto, 1998), as suitably equipped for the situation of an agent exploring a musical environment in realtime. Whilst the research presented here is hardly conclusive, it may stimulate consideration of the potential benefits and pitfalls of RL approaches.

In prior work, Franklin and Manfredi (Franklin & Manfredi, 2002) studied actor-critic reinforcement learning in a jazz improvisation context, generating notes and rating actions with respect to a basic hard-coded jazz music theory. RL has also been explored in the OMax research project at IRCAM (`http://recherche.ircam.fr/equipes/repmus/OMax/`). In the closing stages of a paper, Assayag et al. (2006) weight links in the Factor Oracle algorithm using discrete state-action RL; the reward signal in online interaction is heightened performer attention to particular materials. An offline mode feeds back the system's generated material to itself with negative reinforcement to increase variety in productions.

To effect learning in RL, a reward signal is necessary to clarify the success of any action choice, given the state. Various RL algorithms exist; the Sarsa(λ) variant utilised herein allows the back-propagation of reward values along the set of recent state-action pairs with falloff of influence controlled by λ. Salient notions of reward for interactive music systems might include quality of anticipation, influence on other parties (perhaps in take-up of material), and direct feedback from participants or observers on their experience. Human or objective fitness functions can also be used, analogously to interactive genetic algorithm's human assessment bottleneck as against programmed criteria. Human feedback is hard to solicit, however; whilst physiological sensors such as GSR or EEG may provide continuous measures of emotional state, their trustworthiness remains to be proven in future investigation. Whilst in other recent work I have explored signals based on 'prediction' ability and a degree of 'consequence', this article considers direct user entry of quality ratings, despite potential concerns on this task being a distractor from playing.

2. A test case for online tuition using Sarsa(λ)

Whilst experiments with rather more complicated state-action spaces and alternative reward signals have been carried out with *Improvagent*[1], a simpler study is presented here in order to provide a stimulus for discussion. In this symbolic (MIDI) system, a user plays the keyboard, and various features are abstracted within a time window of three seconds modelling the

[1]Paper to be presented at ICMC 2008

perceptual present; the state is updated once per second. This external data is the current environment state to which the system must respond with an action. To keep the setting manageable, three features are extracted (log density, pitch variance and log latest event time) and five bands allowed to partition their values. Five behaviours are defined (0=silence, 1-4 are small imitative/processing algorithms exploiting collected note events from the current window). The state-action space thus has 625 (5^4) values to determine for a performance policy. The performer enters rewards at any time on the computer keyboard, using one key for positive reward and one for negative reward, reminiscent of Al Biles' online IGA for GenJam.

Testing the system revealed how closely the coverage of states depended on fine representational decisions; for instance, the values particular features take on can be decidedly non-uniform. Even after scaling adjustments, only 20% of states were touched in a typical ten minute interaction session. In Sarsa(λ) training, a ten minute session would give only 600 cases; an offline process of re-running these cases in internal rehearsal can be used to refine estimates to speed up convergence. Without higher level musical goals and extensive long-term training, it was hard to tell how long-term interaction quality was affected; but it was possible to stamp out certain aberrant behaviours, and encourage favoured modes, by feeding back negative and positive reward.

3. Discussion

Reinforcement learning experts have debunked some common myths concerning RL schemes, such as the slowness of training, the divergence under function approximation and that a strict MDP is required (`http://neuromancer.eecs.umich.edu/cgi-bin/twiki/view/Main/MythsofRL`). It must be admitted, however, that human performers can be wilfully non-deterministic, taking a different choice given the same situation; it is hoped that the statistics of their own policies can be tractably modelled. Whilst as noted by Belinda Thom (Thom, 2003) any given musical interaction may offer an extreme case of sparse data, the extensive practice regimes of thousands of hours carried out by expert human musicians may be offered as a counter-example to illustrate the amount of training data potentially required (Deliège & Sloboda, 1996). A pertinent question is how best to scale up small systems into larger systems worthy of the investment of such time, and how to train in non-interactive situations (from MIDI file data, say) in readiness for real interaction.

The choice of representation is critical as ever, and it may be productive to consider more closely how human musicians manage to encode and recall musical data, that is, to employ psychological data on human learning and memory. Chunking of data (storage as manageable chunks of around three notes at a time with context in a memory record) may be an ecologically valid tactic for keeping dimensionality low.

4. Conclusions

Reinforcement learning has potential for learning in interactive music systems, but with many outstanding issues concerning nature (bootstrapping) and nurture (learning in an environment) and associated representations and modelling. There are also questions of evaluation for the systems themselves, which can be approached on a number of levels, from participant experience, to audience and critical response, alongside more objective technical criteria.

References

Assayag, G., Bloch, G., Chemillier, M., Cont, A., & Dubnov, S. (2006). OMax brothers: a dynamic topology of agents for improvization learning. *AMCMM '06: Proceedings of the 1st ACM workshop on audio and music computing multimedia* (pp. 125–132).

Collins, N. (2006). *Towards autonomous agents for live computer music: Realtime machine listening and interactive music systems*. Doctoral dissertation, University of Cambridge.

Collins, N. (2007). Musical robots and listening machines. In N. Collins and J. d'Escrivan (Eds.), *Cambridge companion to electronic music*, 171–84. Cambridge: Cambridge University Press.

Deliège, I., & Sloboda, J. (1996). *Musical beginnings: Origins and development of musical competence*. New York: Oxford University Press,.

Franklin, J. A., & Manfredi, V. U. (2002). Nonlinear credit assignment for musical sequences. *Second international workshop on Intelligent systems design and application* (pp. 245–250).

Hamanaka, M., Goto, M., Asoh, H., & Otsu, N. (2003). A learning-based jam session system that imitates a player's personality model. *IJCAI: International Joint Conference on Artificial Intelligence* (pp. 51–58).

Rowe, R. (2001). *Machine musicianship*. Cambs, MA: MIT Press.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Thom, B. (2003). Interactive improvisational music companionship: A user-modeling approach. *User Modeling and User-Adapted Interaction Journal*, *13*, 133–77.

Modeling Celtic Violin Expressive Performance

Rafael Ramirez, Alfoso Perez, Stefan Kersten {RAFAEL,APEREZ,SKERSTEN}@IUA.UPF.EDU
Universitat Pompeu Fabra, Ocata 1, 08003 Barcelona, Spain

David Rizo, Placido Roman, Jose M. Inesta {DRIZO,PROMAN,INESTA}@DLSI.UA.ES
Departamento de Lenguajes y Sistemas informticos, Universidad de Alicante, Spain

Abstract

Professional musicians intuitively manipulate sound properties such as pitch, timing, amplitude and timbre in order to produce expressive performances of particular pieces. However, there is little explicit information about how and in which musical contexts this manipulation occurs. In this paper we describe an inductive logic programming approach to modeling the knowledge applied by a violinist when performing a celtic piece in order to produce an expressive performance.

1. Introduction

In this paper we describe an inductive logic programming approach to investigate how skilled musicians express and communicate their view of the musical and emotional content of musical pieces and how to use this information in order to generate expressive performances. Instead of manually modelling expressive performance and testing the model on real musical data, we use sound analysis techniques based on spectral models [4] for extracting high-level symbolic features from the recordings, and let a computer use inductive logic programming techniques to automatically discover performance patterns from real performances. We study deviations of parameters such as timing and amplitude in the context of celtic violin music.

By applying inductive logic programming techniques we are able to express the performance regularities as first order logic rules, and introduce background knowledge in the learning process. The increased expressiveness of first order logic not only provides a more elegant and efficient specification of the musical context of a note, but it provides a more accurate predictive model [3] The possibilty of involving background knowledge in the learning process is of special interest in musical applications where often there is musical background knowledge available.

2. Expressive performance modeling

Training data. In this work we are focused on Celtic jigs, fast tunes but slower that reels, that usually consist of eighth notes in a ternary time signature, with strong accents at each beat. The training data used in our experimental investigations are monophonic recordings of nine Celtic jigs performed by a professional violinist. Apart from the tempo (they played following a metronome), the musician was not given any particular instructions on how to perform the pieces.

Musical Analysis. It is widely recognized that expressive performance is a multi-level phenomenon and that humans perform music considering a number of abstract musical structures. In order to provide an abstract structure to our performance data, we decided to use Narmours theory [2] and tonal analysis [5] to analyse the performances.

Note descriptors. We characterize each performed note by a set of features representing both properties of the note itself and aspects of the musical context in which the note appears. Information about the note includes note pitch and note duration, while information about its melodic and harmonic context includes the relative pitch and duration of the neighboring notes (i.e. previous and following notes), classification from the harmonic point of view (e.g. passing tone, neighbor tone, harmonic note), as well as the Narmour structures to which the note belongs. The note's Narmour structures are computed by performing the musical analysis according to Narmour's theory. Thus, each performed note is contextually characterized by the tuple

(Pitch, Dur, MetrStr, PrevPitch, PrevDur, NextPitch, NextDur, Nar1, Nar2, Nar3)

Learning Algorithm. We used Tildes top-down decision tree induction algorithm [1]. Tilde can be considered as a first order logic extension of the C4.5 decision tree algorithm: instead of testing attribute values at the nodes of the tree, Tilde tests logical predicates. This provides the advantages of both propositional decision trees (i.e. efficiency and pruning techniques) and the use of first order logic (i.e. increased expressiveness).

We apply the learning algorithm with the following target predicates: `duration/3`, `energy/3`, and `alteration/3`. (where `/n` at the end of the predicate name referes to the predicate arity, i.e. the number of arguments the predicate takes). Each target predicate corresponds to a particular type of transformation: `duration/3` refers to duration transformation, `energy/3` to energy transformation, and `alteration/3` refers to melody alteration.

For each target predicate we use as example set the complete training data specialized for the particular type of transformation, e.g. for `duration/3` we used the complete data set information on duration transformation (i.e. the performed duration transformation for each note in the data set). The arguments are the musical piece, the note in the piece and performed transformation.

We use (background) predicates to specify both note musical context and background information. The predicates we consider include `context/6`, `narmour/2`, `succ/2` and `member/3`. Predicate `context/8` specifies the local context of a note. i.e. its arguments are *(Pitch, Dur, MetrStr, PrevPitch, PrevDur, NextPitch, NextDur)*. Predicate `narmour/2` specifies the Narmour groups to which the note belongs. Its arguments are the note identifier and a list of Narmour groups. Predicate `succ(X,Y)` means `Y` is the successor of `Y`, and Predicate `member(X,L)` means `X` is a member of list `L`. Note that `succ(X,Y)` also mean `X` is the predecessor of `Y`. The `succ(X,Y)` predicate allows the specification of arbitrary-size note-context by chaining a number of successive notes: $succ(X_1, X_2)$, $succ(X_2, X_3), \ldots$, $succ(X_{n-1}, X_n)$ where X_i $(1 \leq i \leq n)$ is the note of interest.

Results. We obtained correlation coefficients of 0.88 and 0.83 for the duration transformation and note dynamics prediction tasks, respectively and we obtained a correctly classified instances percentage of 86% for ornamentation prediction. These numbers were obtained by performing 10-fold cross-validation on the training data. The induced models seem to capture accurately the expressive transformations the musician introduces in the performances. Figure 1 contrasts the note duration deviations predicted by the model and the deviations performed by the violinist. Similar results were obtained for the dynamics model.

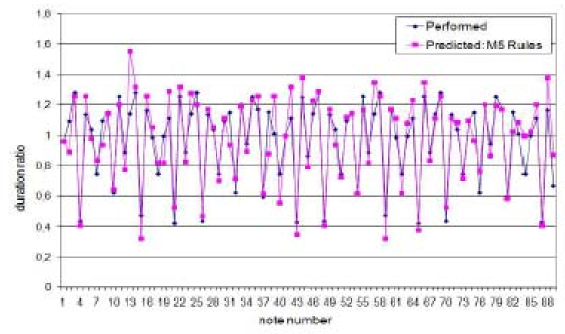

Figure 1. Note deviation ratio for a tune with 89 notes. Comparison between performed and predicted by the model

3. Conclusions

This paper describes an inductive logic programming approach to modeling the knowledge applied by a musician when performing a score in order to produce an expressive performance. Our objective has been to obtain a computational model which predicts how a particular note in a particular context should be played In order to induce the expressive performance model, we have applied an inductive logic programming algorithm to data extracted from audio recordings and information about the context in which the data appear.

Acknowledgments. This work is supported by project TIN2006-14932-C01.

References

[1] Blockeel, H., De Raedt, L., and Ramon. J. (1998). Top-down induction of clustering trees. In ed. J. Shavlik, editor, Proceedings of the 15th International Conference on Machine Learning, pages 53-63, Madison, Wisconsin, USA, Morgan Kaufmann.

[2] Narmour, E. (1990). The Analysis and Cognition of Basic Melodic Structures: The Implication Realization Model. University of Chicago Press.

[3] Ramirez, R., Hazan, A. (2006). A Tool for Generating and Explaining Expressive Music Performances of Monophonic Jazz Melodies, International Journal on Artificial Intelligence Tools, 15(4), pp. 673-691

[4] Serra, X. and Smith, S. (1990). Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition, CMJ 14(4).

[5] Illescas, P., Rizo, D., Iesta, J.M. (2007) Harmonic, melodic, and functional automatic analysis, Proceedings of ICMC07.

Identifying Cover Songs Using Normalized Compression Distance

Teppo E. Ahonen, Kjell Lemström {TEAHONEN,KLEMSTRO}@CS.HELSINKI.FI
Department of Computer Science, University of Helsinki

Abstract

Normalized Compression Distance (NCD) is a distance metric that approximates Kolmogorov-complexity-based information distance by using a standard data compression algorithm. It has been successfully used for clustering symbolic music data. In this paper we use NCD for identifying cover versions by calculating the similarity between chord sequences estimated from audio files.

1. Introduction

A cover song is an alternative rendition of a previously recorded song. In popular music a cover song is usually very different from the original song: the cover version may vary, for instance, in tempo, key, arrangement or the language of the vocals.

Thus, identifying cover songs automatically is a rather difficult task. The features that are used for the identification must be robust: the so-called low-level audio signal features do not remain unchanged between the original and the cover version. The cover song identification is done by comparing the mid-level representations of the songs. Usually this means estimating the chord sequences from the songs and then calculating the distance between the sequences using methods such as edit distance or dynamic time warping. Automatic cover song identification is important not only because it can be used for managing large collections of digital audio, but also because identifying cover songs yields important information on how musical similarity can be measured and modelled.

Normalized Compression Distance, or NCD, is a distance metric that is based on Kolmogorov complexity. NCD is universal and requires no background information of the subject domain. We propose a method that uses NCD to distinguish the cover songs by calculating the similarity between chord sequences.

2. Chord estimation

Estimating the chord sequences in audio data is a well-researched area. The common process for chord estimating consists of two steps: pre-processing the audio into a feature vector representation and approximating the most likely chord sequence from the vectors by a Hidden Markov Model (Papadopoulos & Peeters, 2007).

First, the audio data is turned into a *chromagram*, twelve-dimensional vectors representing the intensity of the twelve pitch classes in a given time slice. There are several ways to do this: a straightforward method is the use of the constant Q transform (Brown, 1991).

Then, the chords are estimated from the chroma vectors using a Hidden Markov Model. The states are the chords, and the observations are the chroma vectors. The initial model parameters can be chosen using different modellings that vary from musical knowledge to cognitive experiments (Papadopoulos & Peeters, 2007). Finally, the model is trained using the expectation-maximization (EM) algorithm and the most likely chord sequence is estimated using the Viterbi algorithm.

3. Brief introduction to NCD

The *Kolmogorov complexity* for a string x is the length of the shortest binary program that produces x as output, denoted $K(x)$ (Chen et al., 2004). The *conditional Kolmogorov complexity* for strings x and y, denoted $K(x|y)$, is the length of the shortest binary program that produces output x given input y. Based on this, the *information distance* $E(x, y)$ is the length of the shortest binary program that outputs x given input y and vice versa. Up to an additive logarithmic term, $E(x, y) = \max\{K(x|y), K(y|x)\}$ (Chen et al., 2004).

Information distance is an absolute distance: it does not consider the lengths of the input strings. The similarity metric, however, should be relative. We normalize the information distance to the range $[0, 1]$. *Nor-*

malized Information Distance (NID) is calculated as follows (Chen et al., 2004):

$$NID(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}. \quad (1)$$

However, Kolmogorov complexity is incomputable in the Turing sense (Cilibrasi & Vitányi, 2005). Although NID cannot be computed, we can approximate Kolmogorov complexity by using any standard lossless data compression algorithm (Cilibrasi & Vitányi, 2005), such as gzip, bzip2 or PPMZ.

Denote $C(x)$ as the length of the string x when compressed using compression algorithm C, and $C(y|x) = C(xy) - C(x)$ as the number of bits in the compressed xy compared to the compressed x.

Using a standard compressor algorithm C to approximate Kolmogorov complexity we can now approximate the NID in equation (1). This is called Normalized Compression Distance and is calculated as follows (Cilibrasi & Vitányi, 2005):

$$NCD(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}.$$

4. Results and conclusions

We implemented an experimental system based on the concepts introduced here. The system was mostly coded in MATLAB using various MIR toolkits, apart from the NCD part that uses the CompLearn toolkit[1]. The chord sequences are estimated from the constant Q transformed vectors using a 24-state HMM, with initial parameters chosen according to the musical knowledge of chord changes as proposed by Bello and Pickens (Bello & Pickens, 2005). Chord sequences are written into text files using a brief format that summarises the chord change into three characters: the first two denote the semitone difference between the root notes of the chords, and the third tells whether there is a change between a major and a minor chord. For example, the chord progression `F G Gm A` would be written as `+2N +2C +2C`.

The experiments were carried out using the `covers80` dataset[2], which is a 160-song collection of 80 original songs and their cover versions. The distances were calculated using two standard compression algorithms: the gzip and bzip2. For each 80 cover song queries an answer set of three most similiar songs was returned. The performance was measured based on these sets. We calculated the number of exact matches (the queries with original song having the smallest distance), the top-3 recognition rate (the number of queries where the original song was found in the answer set) and the mean of average precisions, MAP (the mean of query precisions emphasizing the rank of the match in the answer set). Results are listed in table 1.

Table 1. Results of the NCD Cover Song Identification test.

Measure	Value range	gzip	bzip2
Match	[0-80]	9	10
Top-3	[0-80]	17	20
MAP	[0-1]	0.156	0.175

The results are not very satisfying, but they do imply that the NCD approach has potential for cover song identification as the NCD is able to identify the cover songs based on the roughly estimated chord sequences. Also, the songs discovered were mostly the same with both compressors, thus implying that NCD works regardless of the selected compressor, although using bzip2 seems to yield slightly better results. Currently, the work is focused on improving the chord estimation and searching for a more suitable notation to represent the chord sequences.

References

Bello, J. P., & Pickens, J. (2005). A robust mid-level representation for harmonic content in music signals. *Proc. ISMIR'05* (pp. 304 – 311). London, UK: Queen Mary, University of London.

Brown, J. C. (1991). Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America*, *89*, 425 – 434.

Chen, X., Li, M., Li, X., Ma, B., & Vitányi, P. M. B. (2004). The similarity metric. *IEEE Transactions on Information Theory*, *50*, 3250 – 3264.

Cilibrasi, R., & Vitányi, P. M. B. (2005). Clustering by compression. *IEEE Transactions on Information Theory*, *51*, 1523 – 1545.

Papadopoulos, H., & Peeters, G. (2007). Large-scale study of chord estimation algorithms based on chroma representation and HMM. *Proc. CBMI '07* (pp. 53 – 60). Paris, France: IRCAM.

[1]http://www.complearn.org

[2]http://labrosa.ee.columbia.edu/projects/coversongs/covers80/

Towards Logic-based Representations of Musical Harmony for Classification, Retrieval and Knowledge Discovery

Amélie Anglade AMELIE.ANGLADE@ELEC.QMUL.AC.UK
Simon Dixon SIMON.DIXON@ELEC.QMUL.AC.UK
Centre for Digital Music, Queen Mary University of London, Mile End Road, London E1 4NS, UK

Abstract

We present a logic-based framework using a relational description of musical data and logical inference for automatic characterisation of music. It is intended to be an alternative to the bag-of-frames approach for classification tasks but is also suitable for retrieval and musical knowledge discovery. We present the first results obtained with such a system using Inductive Logic Programming as inference method to characterise the Beatles and Real Book harmony. We conclude with a discussion of the knowledge representation problems we faced during these first tests.

1. Introduction

Most of the automatic music classification systems (performing e.g. genre classification) for audio data are based on the so-called "bag-of-frames" approach. In this approach, feature vectors (typically spectral features such as MFCC) are computed over short frames. Then a classifier computes the global distribution or average values of these vectors over the whole piece or passage for each class. However this statistical method presents several limitations. For instance it has been shown that contrary to the bag-of-frames assumption the contribution of a musical event to the perceptual similarity is not proportional to its statistical importance (rare musical events can even be the most informative ones) (Aucouturier et al., 2007). Moreover the bag-of-frames approach ignores temporal organisation of the acoustic signal. However when comparing pieces from similar genres or passages of a same song it is crucial in the retrieval process to use sequences and not only average values or global statistical distributions of features over a whole passage or piece (Casey & Slaney, 2006).

We believe a logic-based representation of the musical events together with logical inference such as Inductive Logic Programming (ILP) of higher level phenomena would overcome these limitations. Indeed temporal relations between musical events are easily expressible in a relational framework. Moreover logical inference of rules allows to take into account all events, even those which are rare. Another advantage of using logical rules is that they are human-readable.

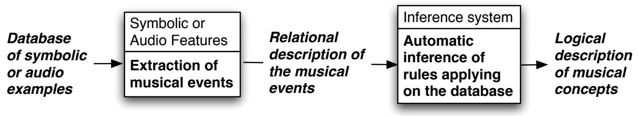
Database of symbolic or audio examples
Symbolic or Audio Features
Extraction of musical events
Relational description of the musical events
Inference system
Automatic inference of rules applying on the database
Logical description of musical concepts

Figure 1. Logic-based framework for musical rule induction

Additionally logical inference has already been successfully used in several music-related projects, for instance to induce rules about popular music harmonisation (Ramirez, 2003), counterpoint (Morales & Morales, 1995) and expressive performance (Widmer, 2003).

2. The Reasoning Framework

Taking the example of these promising applications of logic to music we intend to build a logic-based reasoning system able to characterize songs for classification or similarity evaluation purposes. An illustration of this system is given in Figure 1. It takes a database of examples to characterise. Notice that these examples can either be audio signals or symbolic examples. These examples are then analysed by a musical event extractor using either symbolic or audio features. Finally the relational description of the examples resulting from this analysis is given to an inference system which derives musical rules that are true for the examples. An example of such an automatically derived rule for the blues genre would be: $12\_bar\_structure(X) \wedge blues\_scale(X) \wedge syncopation(X) \Rightarrow blues(X)$

We imagine three kinds of applications for this system:

1. **Classification and retrieval** of pieces similar to a given set of examples, for instance for recommender systems.

2. **Musical knowledge discovery**: some of the derived rules might describe new interesting musical phenomena.

3. **Query in musical databases**: finding music that matches a pattern/description expressed by the user in the form of a logic formula (either defined by hand or a rule previously derived by the system) so without any audio or score example.

3. First Results

We ran a first set of experiments using the ILP system Aleph (Srinivasan, 2003) which is based on Inverse Entailment. Our objective was to characterize the harmony of the 180 songs featured on the Beatles' studio albums (containing a total of 14,132 chords) and of 244 jazz standards from the Real Book (24,409 chords). More precisely we focused on characterising chord sequences of length 4 in these corpora. To avoid the event extraction step we analysed manually annotated chord data[1] containing information about the root, bass note and component intervals (relative to the root) of the chords. We pre-processed these data to obtain high-level information such as chord category (e.g. major, minor, suspended), degree (e.g. tonic, dominant) and intervals between chords, before passing them to an ILP system which extracted the harmony rules underlying them. It generated 3667 harmony rules characterising and differentiating the Real Book songs and the Beatles music. Examples of learned rules are given in Table 1. Encouragingly some of these rules cover well-known jazz or pop harmonic patterns such as ii-V-I-IV and the "turnaround" pattern I-VI-II-V (for the Real Book), and the I-IV-I-V pattern (for the Beatles). It was also possible to identify patterns specific to the Beatles such as the extensive use of major chords (the predominant maj-maj-maj-maj pattern) and the cyclic patterns (e.g. I-IV-V-I-IV...) characterising their early compositions. More details on these experiments can be found in (Anglade & Dixon, 2008).

4. Discussion

Even though these first results are encouraging we identified some knowledge representation problems arising when using ILP as inference technique. In ILP both the concept and the vocabulary to describe it (or background knowledge) needs to be defined in advance. Thus, two different vocabularies result in different descriptions of the concept. The process of concept characterisation is interactive: to end up with an interesting theory we might need to manually refine the vocabulary several times. For instance for the Beatles in our final attempt we used the concept "chord sequence (of length 4) in the Beatles" (`chord_seq/4`) and asked the system to describe it in terms of "chord category sequence" (`category_seq/8`), "root interval sequence" (`rootInterval_seq/7`), etc. But our earlier attempts at using only low level concepts (e.g. using independent descriptions of each chord, linked only by a "predecessor" predicate) failed to provide meaningful descriptions of the target concept. However it seems that as we refine the vocabulary we inevitably reduce the problem to a pattern matching task and not to a pattern discovery task as we intend to, allowing us to validate or refute hypotheses about the concept but not to make any really novel (knowledge) discoveries.

[Rule 16][Pos cover=7.86%]: chord_seq(A,B,C,D):-rootInterval_seq(A,B,C,D,[perf,4th],[perf,4th],[perf,4th]).
[Rule 25][Pos cover=4.09%]: chord_seq(A,B,C,D):-category_seq(A,B,C,D,min,dom,min,dom).
[Rule 57][Pos cover=2.01%]: chord_seq(A,B,C,D):-bassInterval_seq(A,B,C,D,[maj,7th],[perf,4th],[perf,4th]).

Table 1. Examples of rules learned on the Real Book data

5. Acknowledgments

This work is part of the OMRAS2 project funded by EPSRC grant EP/E017614/1.

References

Anglade, A., & Dixon, S. (2008). Characterisation of harmony with Inductive Logic Programming. *Under review for ISMIR 2008.*

Aucouturier, J.-J., Defreville, B., & Pachet, F. (2007). The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. *JASA*, *122*, 881–91.

Casey, M., & Slaney, M. (2006). The importance of sequences in musical similarity. *Proc. of ICASSP 2006.* Toulouse, France.

Morales, E., & Morales, R. (1995). Learning musical rules. *Proc. of the IJCAI-95 Int. Workshop on Artificial Intelligence and Music.* Montréal, Canada.

Ramirez, R. (2003). Inducing musical rules with ILP. *Proceedings of ICLP2003.* Springer-Verlag (LNCS).

Srinivasan, A. (2003). The Aleph Manual (version 4).

Widmer, G. (2003). Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *AI*, *146*, 129–148.

[1] RDF descriptions of the Real Book chords (available on http://chordtranscriptions.net/) and of Harte's transcription of the Beatles (available on request)

Metropolis-Hastings Sampling in a FilterBoost Music Classifier

Balázs Kégl BALAZS.KEGL@GMAIL.COM
Laboratoire de l'Accélérateur Linéaire, Université Paris-Sud 11, FRANCE

Thierry Bertin-Mahieux BERTINMT@IRO.UMONTREAL.CA
DIRO, University of Montreal, Montreal, QC, CANADA

Douglas Eck DOUGLAS.ECK@UMONTREAL.CA
DIRO, University of Montreal, Montreal, QC, CANADA

Abstract

Rejection sampling is a useful technique for performing supervised learning on training sets too large to be learned in their entirety. FilterBoost is a recent extension to AdaBoost which uses rejection sampling in an online learning framework and has been shown to work for automatic tagging of music. In this paper we improve on FilterBoost by adding Metropolis-Hastings sampling, thus allowing the algorithm to focus on hard-to-classify examples. We describe how our knowledge of artist-level similarity can be used effectively in a Metropolis-Hastings framework and demonstrate a significant increase in classification accuracy over standard FilterBoost.

1. Introduction and Previous Work

Rejection sampling is a useful technique for performing supervised learning on training sets too large to be learned in their entirety. In particular, FilterBoost is a recent extension to AdaBoost based on that concept: an oracle gathers data that is later accepted by a filter by a probability related to classification error, thus forcing the model to focus more attention to hard-to-classify data points. This method has been shown to work for automatic tagging of music (Bertin-Mahieux et al., 2008). However, this approach does not take advantage of any structure in our data. For example, if we have a similarity measure for data points we can improve performance by increasing our probability to sample near-neighbors to a difficult point. We possess several similarity measures for working with recorded music. Perhaps the simplest is to treat songs coming from the same album or same artist as being similar. This is a strong assumption, but more refined measures of similarity can also be used.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1.1. FilterBoost and Metropolis-Hasting Sampling

Much attention has been paid to the automatic description of audio using a fixed vocabulary of words (Eck et al., 2008; Turnbull et al., 2008). In previous work we demonstrated that the AdaBoost large-margin ensemble learner performs well on this task (Bergstra et al., 2006). The challenge therefore is the scaling problem: our current music collection contains more than $120K$ songs, and a commercial system has millions of them. How can we apply such a classifier to what is, for practical purposes, an infinitely large data set?

A recent extension of AdaBoost was presented at NIPS 2007 (Bradley & Schapire, 2008). FilterBoost samples <train,target> pairs from a data store (or "oracle") assumed to be infinite, but only learns from a candidate pair if it cannot already classify it. Otherwise it rejects that point and samples another one. Bradley et. al showed tremendous speed increases and memory savings with no loss in performance. By default, FilterBoost considers the training data to be i.i.d. However this is not the case for recorded music where there is strong inter-album and inter-artist similarity. One standard way to take advantage of such structure is via the Metropolis-Hastings algorithm (Bishop, 2006) which is used to sample from a complex probability distribution, and is a special case of the wide framework of Markov Chain Monte Carlo. From a point $z^{(\tau)}$

we draw a sample z^* with probability

$$A(z^*, z^{(\tau)}) = \min\left(1, \frac{\tilde{p}(z^*)q_k(z^{(\tau)}|z^*)}{\tilde{p}(z^{(\tau)})q_k(z^*|z^{(\tau)})}\right) \qquad (1)$$

If the sample z^* is refused, we keep $z^{(\tau)}$.

In our standard FilterBoost model, the oracle samples equally from positive and negative examples and rejects based on how easily the current strong learner classifies. Then the filter looks how easily the current strong learner classifies it. The filter accepts the example at iteration t with probability

$$q_t(x, l) = 1/(1 + e^{lg_t(x)}) \qquad (2)$$

where $lg_t(x)$ is positive if x well-classified, negative otherwise. We train 3000 weak learners using FilterBoost, weak learners being single stumps. Each iteration of FilterBoost, we gather a mini batch of size 3000, and we select 50 single stump using classical AdaBoost. Then we gather another 3000 examples to evaluate the weight of those 50 new weak learners.

We incorporate Metropolis-Hastings into FilterBoost as a direct replacement for the simple rejection filter. The oracle samples from the same artist with probability q_t from equation 2, and randomly from the database with probability $1 - q_t$. The probability of an example is also computed using equation 2.

2. Experiment and Results

Our data come from social tags obtained from the Last.fm website using their AudioScrobbler service. Matched audio is taken from a lab collection of more than $120K$ audio files spanning more than 5600 artists. Audio features include MFCCs, autocorrelation coeffs. and Constant-Q coeffs. summarized (mean and standard deviation) every five seconds. For each tag, artists are ordered by the (normalized) tag frequency. For each tag, the songs from the top 10 artists are considered positive examples, following artists (up to a thousand) are ignored because we are unsure about them. Nonetheless, we can reasonably expect these examples to be usually positive. The rest of the artists in the database are used as negative examples. See (Eck et al., 2008) for details.

Figure 1 gives results for the tag *Grunge*. We see a significant improvement for the new approach on negative examples, and only a small decrease in accuracy for the positive and ignore examples. We conclude that the Metropolis-Hastings algorithm has helped FilterBoost focus on hard (generally negative) examples, thus improving performance significantly.

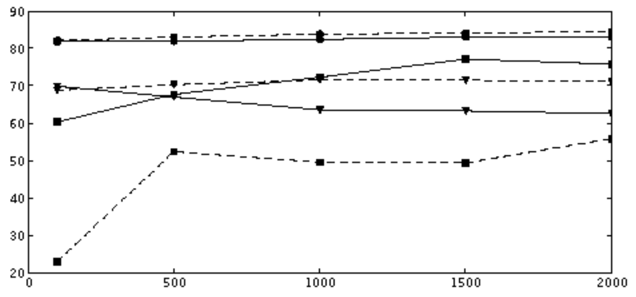

Figure 1. Accuracy by the number of weak learners for tag *Grunge*. The thick line uses Metropolis-Hastings, the dotted line our older method. The held out positive examples are circles, the ignore examples are triangles, and the held out negative examples are squares.

3. Conclusion and Ongoing Work

Previously (Bertin-Mahieux et al., 2008) we demonstrated that FilterBoost works significantly better than AdaBoost at this task. Here we demonstrate that a Metropolis-Hastings rejection policy outperforms a simple error-driven rejection policy. Our current measure of similarity is crude: songs by the same artist are similar. In future work (probably before the workshop) we will incorporate more sophisticated similarity measures which take into account inter-artist similarity . Finally we note that these results are preliminary; more complete results will be available at the time of the workshop.

References

Bergstra, J., Casagrande, N., Erhan, D., Eck, D., & Kégl, B. (2006). Aggregate features and AdaBoost for music classification. *Machine Learning*, *65*, 473–484.

Bertin-Mahieux, T., Eck, D., Maillet, F., & Lamere, P. (2008). Autotagger: a model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*. (to appear).

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer Verlag.

Bradley, J. K., & Schapire, R. (2008). Filterboost: Regression and classification on large datasets. In *Nips 2007*.

Eck, D., Lamere, P., Bertin-Mahieux, T., & Green, S. (2008). Automatic generation of social tags for music recommendation. In *Nips 2007*.

Turnbull, D., Barrington, L., Torres, D., & Lanckriet, G. (2008). Semantic annotation and retrieval of music and sound effects. *IEEE Trans. on Audio, Speech & Lang. Proc.*

An Ensemble-Based Approach for Automatic Hierarchical Music Genre Classification

Carlos N. Silla Jr. CNS2@KENT.AC.UK
Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK

Alex A. Freitas A.A.FREITAS@KENT.AC.UK
Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK

Abstract

In this paper we propose and evaluate an ensemble-based approach for the task of automatic hierarchical music genre classification. The ensemble method consists of employing a hierarchical top-down strategy using information from three different parts of the music signal. The experiments were performed on the Latin music database that contains 3,160 music pieces from 10 Latin genres organized in a two-level hierarchy. The proposed method has show improvements in the classification of the first level of the hierarchy.

1. Introduction

The task of music genre classification plays an important part in any Music Information Retrieval (MIR) System. Studies concerning user behavior of MIR systems show that the musical genre is the second most used information for retrieval of music pieces (Downie & Cunningham, 2002). The task of automatically assigning a music genre based on the music content (i.e. the music signal), has been motivated by the work of (Tzanetakis & Cook, 2002). Since then, the task of automatic music genre classification has received a growing interest by the MIR research community but relatively less work has considered the task of automatic hierarchical music genre classification.

According to (Sun et al., 2003; Freitas & de Carvalho, 2007) there are four possible approaches for hierarchical classification problems. In this work we use the multi-class top-down (also known as local-learning) approach that consists of creating a classifier for every parent node of the class tree, so that each classifier has to discriminate among the sub-classes associated with its child nodes. Therefore a classifier is trained for each node in the hierarchy that is not a leaf node. After a tree of classifiers has been built in the training phase, during the testing phase each new test example (unseen during training) is classified in a top-down fashion as follows. First, the example is assigned a first-level (most generic) class by the classifier ensemble at the root node (at class level 0). Then the example is sent to the root node's child node (at class level 1) whose class was predicted by the root's classifier ensemble. Then the example is assigned a second-level (more specific) class by that classifier ensemble at level 1. If there were more than two class levels this top-down process would be recursively repeated until a leaf class was assigned to the example.

2. Automatic Hierarchical Classification of Music Genres

The music genre classification system proposed in this work is composed of two main phases: feature extraction and ensemble-based top-down multi-class classification. For the baseline approach features are extracted from the middle of the audio signal considering a 30-second segment. For the extraction of features from the music segments, the MARSYAS[1] framework was employed. The MARSYAS framework implements the original feature set proposed by (Tzanetakis & Cook, 2002) that consists of features related to the timbral texture, to the beat and to the pitch. In total 30 features are calculated for each song.

To perform the ensemble-based top-down multi-class classification we extract features from three different parts of the song (beginning, middle and end) and combine them using majority voting (VOTE-TD) or the Max Rule (MAX-TD); In VOTE-TD the final class label is given by the majority voting of the class outputs. While, in MAX-TD the final class label is given by the class with the lowest distance of all classifiers.

[1] Available at: http://marsyas.sourceforge.net

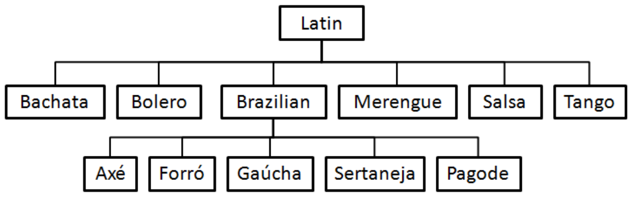

Figure 1. Latin Music Database Class Hierarchy

The main difference from our previous work (Silla Jr. et al., 2007a) is that in this work we combine our previously-proposed ensemble method for flat classification with the hierarchical top-down approach. I.e. at each non-leaf classifier there is an ensemble based classifier.

3. Experiments

The main objective of our experiment was to evaluate the performance of the top-down ensemble-based method for the task of automatic hierarchical music genre classification. For this purpose we have used the Latin Music Database (Silla Jr. et al., 2007b) which contains music samples from ten different Latin genres. We have selected 300 samples from each genre and performed 10-fold cross-validation. The hierarchy used in the experiments is show in Figure 1. The underlying base classifier used was the k-NN (k = 3) and for the standard Top Down (TD) approach, the middle segment was used.

Table 1 presents the predictive accuracy results achieved by the different methods. At the first level of the class hierarchy both ensemble methods achieved higher results than the standard top down approach. At the second level of the hierarchy the ensemble did not outperform the standard approach. This leads to the conclusion that at the first level the ensemble was successful in improving the classification of genres other than Brazilian, and therefore it has not improved the accuracy of the second level. This is one of the drawbacks of the Top-Down approach where errors made at the higher levels of the hierarchy will be propagated through the classification process.

4. Concluding Remarks

In this work we presented an ensemble–based approach for automatic hierarchical music genre classification using a multi-class top-down approach. The experiments of the presented method were compared against a standard hierarchical multi-class top-down classifier. The results suggest that the proposed method achieved better accuracy on the first level of the hierarchy and similar results to the second level.

Table 1. Predictive accuracy at each level of the hierarchy.

Hierarchy	TD	MAX-TD	VOTE-TD
Level 1	75.83± 2.7	80.86± 1.82	79.49± 1.93
Level 2	42.19± 4.0	41.39± 3.11	40.12± 4.46

As future work we pretend to join to the Latin Music database with other existing databases to have a richer hierarchy, in which we believe the proposed method would perform better than the standard approach. Another possibility to improve the proposed method is, instead of extracting features from different parts of the music, extracting different features sets and combining them.

Acknowledgment

The first author is financially supported by the CAPES Brazilian Agency, process number 4871-06-5.

References

Downie, J. S., & Cunningham, S. J. (2002). Toward a theory of music information retrieval queries: System design implications. *Proc. Third Int. Conf. on Music Inform ation Retrieval* (pp. 299–300).

Freitas, A. A., & de Carvalho, A. C. P. L. F. (2007). *Research and trends in data mining technologies and applications*, chapter A Tutorial on Hierarchical Classification with Applications in Bioinformatics, 175–208. Idea Group.

Silla Jr., C. N., Kaestner, C. A. A., & Koerich, A. L. (2007a). Automatic music genre classification using ensemble of classifiers. *IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1687–1692).

Silla Jr., C. N., Kaestner, C. A. A., & Koerich, A. L. (2007b). The latin music database: Uma base de dados para a classificação automática de gêneros musicais. *11th Brazilian Symposium on Computer Music* (pp. 167–174).

Sun, A., Lim, E.-P., & Ng, W.-K. (2003). Performance measurement framework for hierarchical text classification. *Journal of the American Society for Information Science and Technology*, *54*, 1014–1028.

Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, *10*, 293–302.

Composer classification using grammatical inference

Jeroen Geertzen J.GEERTZEN@UVT.NL
Menno van Zaanen MVZAANEN@UVT.NL
ILK / Dept. of Communication & Information Sciences, Tilburg University, Tilburg, The Netherlands

Abstract

We present an approach to automatic composer recognition based on learning recurring patterns in music by grammatical inference. These patterns are more flexible than, for instance, commonly used Markov chains. The induced patterns are subsequently used in a regular-expression based classifier. We show that this approach yields promising classification results and allows investigation of induced composer-typical sequential structure.

1. Introduction

Where the automatic classification of music genre has received considerable attention recently, composer classification, being a more specific task, has not.

For classifying on genre or composer, supervised machine learning is commonly used, in which an algorithm learns a classification model of how features are related to classes based on labelled training samples. In most studies the features used tend to be local, i.e. the features predominantly describe events in rather short time intervals (e.g. tempo changes). Global features are usually statistical measures of musical pieces, such as the distributions of intervals, certain harmonics, tempo, etc. (e.g. in Backer et al., 2005).

When we aim to classify music by composer, we are essentially looking for recurring patterns of features in musical data that are typical to a specific composer. In order to describe what is happening over a span of time, several researchers have used probabilistic methods, notably Markov chains. A n-th order Markov chain bases the probability for a symbol to occur on the last n symbols. Based on the occurrence of unique or frequent Markov chains, a classifier can be built. However, such models do not allow the capture of more complex sequences.

We propose an approach to automatic composer classification that is based on grammatical inference (GI). GI is a branch of unsupervised machine learning that aims to find underlying structure of symbolic sequential data. Contrary to Markov chains, the sequences that are learned may have variable length and may be non-contiguous.

2. Approach

Music has characteristics relating to several aspects, such as harmony (i.e. intervals), melody, and rhythm. In our approach, we decompose the music along the latter two dimensions.[1]

In each of the dimensions, we look for patterns that characterize the music for that particular dimension, and formulate composer classification as a similarity search in a 2-dimensional vector space. For the induction of patterns, we use GI to obtain typical phrases or patterns in an unsupervised manner. We subsequently use these patterns as features in a supervised classification algorithm to automatically classify musical pieces by composer. This approach is depicted in Figure 1.

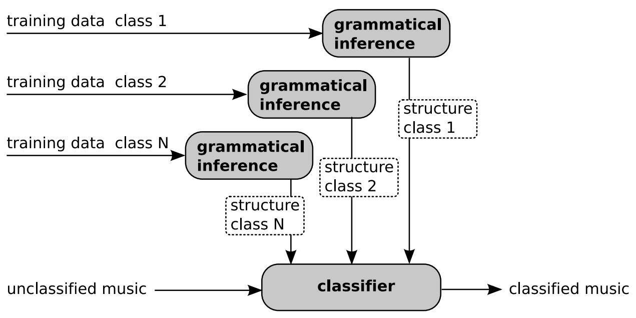

Figure 1. Overview of the GI component in classification.

The GI component in the system is realized by Alignment-Based Learning (ABL) (van Zaanen, 2002). ABL is a generic GI framework that has been successfully applied in natural language processing. It induces structure from sequences by aligning them. Based

[1]There are more dimensions that could be distinguished, but those will be included in future work.

on interchangeable subsequences, the data is generalized. For instance, given the two following melodic sequences:

```
d c d [ e f# ] g e
d c d [ d e ] g e
```

the alignment learning of ABL induces the pattern `d c d X g e`, in which `X` may be any substitutable part. In a similar way, ABL is applied to sequences, each of which representing a piece for a particular dimension.

The supervised classification algorithm determines, for each dimension, the phrases and patterns that are typical to each composer. Subsequently, these global patterns are applied as regular expressions to an unseen piece. This gives for each composer a measure of structural similarity with the piece. The composer with the highest similarity is selected.

3. Data

GI approaches are inherently symbolic and we do not want to consider stylistic aspects related to performance. Therefore, we extracted symbolic representations from music in humdrum **kern format (Huron, 1997). The melodic sequences encode relative pitch changes by means of number of semitones and the direction (upwards or downwards); the rhythmic sequences encode relative meter changes in tempo difference and direction (quicker or slower).

To look for recurring patterns of features in musical data *that is typical to a specific composer*, we compiled two datasets[2] of composers from the same musical era with the same type of musical piece: a baroque dataset with preludes from Bach (42) and Chopin (24), and a classic dataset with quartet pieces from Beethoven (70), Haydn (213), and Mozart (82).

4. Experimental results

The performance of the GI based approach was compared to that of a 2nd order Markov model approach (MM-2) and evaluated using leave-one-out cross-validation.[3] The performance is measured by error rate, which is calculated by dividing the number of incorrectly classified musical pieces by the total number of pieces in the test set.

The resulting scores are given in Table 1. Classification was performed both with a similarity search in a 2-dimensional vector space (joint) and for each dimension in isolation (melody, rhythm).

Table 1. Error rate for both datasets with both approaches.

Dataset	*Dimension*	*ABL*	*MM-2*
BAROQUE	MELODY	19.8 ±0.3	29.1 ±0.4
	RHYTHM	22.5 ±0.6	32.4 ±0.7
	JOINT	19.9 ±0.2	29.0 ±3.6
CLASSIC	MELODY	23.6 ±0.7	34.8 ±2.1
	RHYTHM	28.8 ±1.2	37.2 ±5.9
	JOINT	21.3 ±1.3	35.1 ±2.8

From the table we can conclude that for both datasets, the GI based approach results in lower error rates than the MM-2 approach. Furthermore, it is interesting to see that the joint model (which learns from both melodic and rhythmic sequences) results for the GI system on the classic dataset in a lower error rate than any of the dimensions in isolation.

The discriminative sequences that are found provide ample opportunity for qualitative style analysis, which is beyond the scope of this paper. To give an idea, the following melodic pattern typically occurs in Chopin's preludes: `b♭ e♭ X b♭ b♭ X b♭ X`, where `X` matches an arbitrary number of notes.

Preliminary results in varying the level of detail of the sequences (e.g. relative pitch versus absolute pitch change) indicate that there is still a lot to gain in looking for the optimum balance between detail in data representation and data sparsity.

5. Conclusions & future work

We present a GI based approach to composer classification with promising results. It uses humanly readable patterns automatically extracted from music. Future research will address the use of other GI algorithms, a further exploration of more elaborate and finer grained feature dimensions, as well as motif extraction.

References

Huron, D. (1997). Humdrum and kern: selective feature encoding. In E. Selfridge-Field (Ed.), *Beyond MIDI: The handbook of musical codes*, 375–401. Cambridge, MA, USA: MIT Press.

Backer, E., van Kranenburg, P. (2005). On musical stylometry: a pattern recognition approach. *Pattern Recognition Letters*, 26, 3, pp. 299–309.

van Zaanen, M. M. (2002). *Bootstrapping structure into language: Alignment-Based Learning*. Doctoral dissertation, University of Leeds, Leeds, UK.

[2]A specification of the datasets can be found at: `http://cosmion.net/jeroen/publications/`.

[3]Leave-one-out was used due to the small amount of training data available.

Training Music Sequence Recognizers with Linear Dynamic Programming

Christopher Raphael CRAPHAEL@INDIANA.EDU
School of Informatics, Indiana University, Bloomington, IN 47408 USA

Eric Nichols EPNICHOL@INDIANA.EDU
Center for Research on Concepts and Cognition, Indiana University, Bloomington, IN 47408 USA

Keywords: machine learning, dynamic programming, harmonic analysis, trellis graph

Abstract

An obvious approach to the problem of harmonic analysis is to represent a musical score as a trellis graph, where layers in the graph correspond to musical slices of time, and each node corresponds to the harmonic analysis of the slice. If the transitions between nodes and the analyses at each node are assigned a score, the analysis problem reduces to finding the best-scoring path through the graph, which can be computed with standard dynamic programming (DP). But how can we set the parameters of the necessary score function? We demonstrate a novel algorithm, Linear Dynamic Programming (LDP), to solve the problem of *learning* the coefficients for a linear score function, based on a hand-labeled harmonic analysis.

1. Introduction

Many music recognition problems seek to explain some aspect of music data as a *sequence* of labels that collectively represent a time-varying analysis. Familiar examples of this view come from problems that deal with both symbolic music representations (e.g. MIDI) as well as audio. For instance, harmonic analysis of symbolic music data produces a label including chord, key, or other harmonic attributes, for each measure or beat of the piece under consideration (Temperley, 2001). Other examples of symbolic music sequence analysis include pitch spelling, instrument fingering, prosodic labeling of melody for musical expression, and decomposition of polyphonic music into voices. In the audio domain, the problem of beat tracking can be expressed as a labeling of each audio frame as either a "beat" or "non-beat." Similar views are often taken of other audio music processing examples such as the signal-to-score problem, computing harmony from audio, chorus detection, score alignment, etc. In fact, the sequence analysis view extends to other music data, such as optical music recognition.

The sequence estimation problem can be solved by expressing each possible label sequence as a path through a state graph. Arc costs in the graph correspond to *a priori* path plausibility, while data scores for nodes in the graph give the agreement between a label and an observation. Given this model, it is simple to compute the best-scoring path using DP. However, an oft-ignored problem faced in applying this methodology is the specification of arc and data scores — the training problem. Hand setting of parameters through trial and error is usually only feasible in small problems. Our research has produced a novel method for the *supervised* training of a DP-based sequence estimator we call *Linear Dynamic Programming.* The method assumes that the arc and data scores are known linear functions of some unknown parameter vector. In this context we seek the parameter value leading to the best performance on our labeled training set. An example is presented for harmonic analysis, though we believe the technique may be broadly applicable.

2. Application to Harmonic Analysis

The problem of *functional harmonic analysis* seeks to partition a piece of music into labeled sections, the labels giving the local *harmonic state.* The label usually consists of a key and a chord symbol such as I, ii, iii, IV, V, vi, vii. For instance, the label (A major, IV) corresponds to the triad (D,F♯,A) built on the fourth scale degree in the key of A major. Several efforts have addressed this problem of automatic harmonic analysis, including (Pardo & Birmingham, 2002), (Raphael

& Stoddard, 2003), and (Temperley, 2001).

We seek to label each measure (or beat) of music with a collection of predefined harmonic labels, appropriately chosen for the music in question. Our recognition approach uses DP to find the best scoring path through the lattice composed by an $S \times N$ array of states, where N is the number of measures or beats in the piece and S is the number of possible harmonic labels we consider. Our cost function consists of two components: a data cost and a path cost. The data cost encourages close agreement between each measure label and the pitches of that measure. The path cost rewards paths that are more musically plausible, independent of the data. Our recognized sequence is then computed as the lattice path that minimizes the sum of these costs.

More explicitly, our cost function, $C_\theta(s)$, is composed as $C_\theta(s) = D_\theta(s) + P_\theta(s)$ where the path, s, is a sequence of labels, $s = s_1, \ldots, s_n$, one for each measure. The data score, $D_\theta(s) = \sum_{n=1}^{N} d_\theta(s_n, x_n)$, is represented as

$$d_\theta(s_n, x_n) = \sum_{i=1}^{3} \sum_{j=1}^{4} \theta_{ij}^{d} \delta_{ij}(s_n, x_n) \qquad (1)$$

where x_n is the collection of pitches in the nth measure and the counts, $\delta_{ij}(s_n, x_n)$, are as follows. Each harmonic label, s_n, divides the possible chromatic pitches into four categories: those that are 1) the root of the chord, 2) in the chord but not the root, 3) in the scale but not the chord, and 4) outside the scale. Similarly, the pitches in a measure are divided into those that begin 1) on the downbeat of the measure, 2) on a beat but not the downbeat, and 3) elsewhere in the measure. In Eqn. 1, $\delta_{ij}(s_n, x_n)$ counts the notes in the nth measure that are in position category i and of chromatic type j. To avoid degeneracies in which families of parameter assignments correspond to essentially identical choices, we further assume $\sum_j \theta_{ij}^{d} = 0$.

The path score $P_\theta(s) = \sum_{n=1}^{N-1} p_\theta(s_n, s_{n+1})$ is the sum of modulation and progression components:

$$p_\theta(s_n, s_{n+1}) = \theta^m M(s_n, s_{n+1}) + \sum_{i=1}^{3} \theta_i^h H_i(s_n, s_{n+1})$$

where $M(s_n, s_{n+1})$ is an indicator function for key change. If $M(s_n, s_{n+1}) = 1$, the $\{H_i\}$ terms act as indicators for various classes of harmonic motion such as progressive and regressive. As above, we assume $\sum_i \theta_i^h = 0$. In total, considering the linear constraints, our parameter θ has dimension 12.

The LDP algorithm divides parameter space into regions corresponding to distinct paths through the trellis. The assumption of linear path scores, along with the trellis structure, render this problem tractable — LDP uses DP itself to partition parameter space. Here, LDP yields a collection of distinct harmonic analyses along with regions in parameter space that produce each analysis. LDP chooses the analysis closest to the training data according to a harmonically motivated path quality measure. Then we select a parameter θ from the center of the associated region.

Using this procedure, we trained our algorithm on the *Grande Valse Brilliante* of Chopin using hand-labeled ground truth. LDP improved our path quality measure from 2203 to 173, starting with a random initial configuration for θ. The resulting configuration corresponded to a total summed symmetric difference (SSD) of 73 between the recognized chord pitch classes and the ground truth chord pitch classes, as well as an SSD of 100 between the two scale sequences. This corresponds to 0.24 chord errors and 0.33 scale errors per measure. We then applied this learned parameter to the Chopin "Minute Waltz", resulting in an SSD of 186 chord pitch class errors (1.33/measure) and 40 scale pitch class errors (0.29/measure). The analysis is available for download as a midi file at www.music.informatics.indiana.edu/papers/aaai08/.

Research is ongoing to improve the LDP algorithm and to test it on a larger collection of musical examples. Although our work to date has focused on harmonic analysis, LDP is applicable to other musical domains; indeed, the algorithm was inspired by difficulties encountered in choosing parameters for a piano fingering task (Kasimi et al., 2007). We hope that LDP will prove useful in many other domains where training data exists in the form of a known sequential labeling.

References

Kasimi, A., Nichols, E., & Raphael, C. (2007). A simple algorithm for automatic generation of polyphonic piano fingerings. *Proc. ISMIR.*

Pardo, B., & Birmingham, W. P. (2002). Algorithms for chordal analysis. *Computer Music Journal*, *26*, 27–49.

Raphael, C., & Stoddard, J. (2003). Harmonic analysis with probabilistic graphical models. *Proc. ISMIR.*

Temperley, D. (2001). *The cognition of basic musical structures.* Cambridge, MA: MIT Press.

Genre Classification of Music by Tonal Harmony

Carlos Pérez-Sancho CPEREZ@DLSI.UA.ES
David Rizo DRIZO@DLSI.UA.ES
Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Spain

Stefan Kersten SKERSTEN@IUA.UPF.EDU
Rafael Ramirez RRAMIREZ@IUA.UPF.EDU
Music Technology Group, Universitat Pompeu-Fabra, Barcelona, Spain

Keywords: genre classification, language models, chord transcription, chord progressions

Abstract

We present a genre classification framework for audio music based on a symbolic classification system. Audio signals are transformed to a symbolic representation of harmony using a chord transcription algorithm, by computing Harmonic Pitch Class Profiles. Then, language models built from a groundtruth of chord progressions for each genre are used to perform classification. We show that chord progressions are a suitable feature to represent musical genre, as they capture the harmonic rules relevant in each musical period or style.

1. Genre classification

Organization of large music repositories is a tedious and time-intensive task for which music genre is an important meta-data. Automatic genre and style classification have become popular topics in Music Information Retrieval (MIR) research because musical genres are categorical labels created by humans to characterize pieces of music and this nature provides the genre meta-data with a high semantic and cultural information to the music items in the collection.

Traditionally, the research domain of genre classification has been divided into the audio and symbolic music analysis and retrieval domains. Nevertheless, some authors have paid attention recently on making use of the best of both worlds. The work by Lidy et al. (Lidy et al., 2007) deals with audio to MIDI transcription in order to extract features from both signals and then combine the decisions of the different classifiers. On the other hand, Cataltepe and coworkers' approach (Cataltepe et al., 2007) is just the opposite: to synthesize audio from MIDI and then analyze both signals to integrate the classifications.

Our proposal is to use tonal harmonic information to distinguish between musical genres. The underlying hypothesis is that each musical genre makes use of different rules that allow or forbid specific chord progressions. As it can be found in (Piston, 1987), some rules that were almost forbidden in a period have been accepted afterwards. Also, it is well known that pop-rock tunes mainly follow the classical tonic-subdominant-dominant chord sequence, whereas jazz harmony books propose different series of chord progressions as a standard.

The goal of this work is to classify digital audio music using a language modelling system trained from a groundtruth of chord progressions, bridging the gap between audio and symbolic by means of a chord transcription algorithm.

2. Chord transcription

The Pitch Class Profile (PCP) measure has been used in automatic chord recognition or key extraction since its introduction by Fujishima (Fujishima, 1999). The perception of musical pitch has two main attributes: height and chroma. Pitch height moves vertically in octaves telling which octave a note belongs to, while chroma tells its position in relation to others within an octave. A chromagram or a pitch class profile is a 12-dimensional vector representation of a chroma, which represents the relative intensity in each of twelve semitones in a chromatic scale. Since a chord is composed of a set of tones, and its label is only determined by the position of those tones in a chroma, regardless of their heights, chromagram seems to be an ideal feature to represent a musical chord.

Table 1. Classification accuracy percentages using different n-gram lengths.

Data set	2-grams	3-grams	4-grams
3 classes	54.4	61.1	61.1
popular vs. jazz	70.9	83.4	83.4
academic vs. jazz	75.0	75.0	75.0
academic vs. popular	56.7	66.7	66.7

In this paper we have obtained the Harmonic Pitch Class Profile (HCPC) by applying the algorithm in (Gómez & Herrera, 2004), which deviates from Fujishima's PCP measure by distributing spectral peak contributions to several adjacent HCPC bins and taking a peaks harmonics into account. The feature vectors are then processed to obtain a symbolic representation of chords in the form of triads. Thus, each song is represented as a string of chord progressions, with an alphabet of 24 symbols (major and minor triads).

3. Language models

We have used a language modelling approach to classify chord progressions (Camastra & Vinciarelli, 2008). The idea is to capture the different use of chord progressions in each style by using n-grams of chords taken from a training set. Thus, a language model is built for each genre in the training set, and these models are evaluated against a new set of songs. These songs are then labelled with the genre of the model with the highest probability of having generated it.

4. Experiments

Two different data sets were used in the experiments. As a training set we used a groundtruth of chord progressions, obtained from a set of 761 symbolic music files belonging to three genres: academic (comprising baroque, classical and romanticism), jazz and popular. For testing the system we used a set of 12 audio files from the same genres, from which we obtained the chord progressions by applying the chord transcription algorithm described in section 2.

Table 1 shows the classification accuracy of the system when using different n-gram sizes and configurations of the data sets (all-against-all and binary classifiers). As expected, better results were obtained when classifying jazz music against the other two styles. This is consistent with the fact that the differences in harmony between these styles are usually bigger than they are between academic and popular music. Also, results tend to improve when using larger n-grams, as they are more capable to capture the typical structures present in each style.

5. Conclusions and future work

We have shown the feasibility of classifying digital audio music by genre using a symbolic classification system. For this, a chord transcription algorithm was used to obtain the chord progressions present in each song. The results show a good performance when classifying between harmonically distant genres, while it is more difficult to distinguish when they make use of more similar chord progressions. Nevertheless, we are convinced that these results can be improved by using a richer chord vocabulary, to better represent the slight differences in harmony that are lost when using only chord triads. Also, a bigger set of audio files will be constructed to deeply test this approach.

Acknowledgments

This work is supported by the Spanish PROSEMUS project (TIN2006-14932) an partially supported by the research programme Consolider Ingenio 2010 (MIPRCV, CSD2007-00018) and the European PASCAL2 Network of Excellence.

References

Camastra, F., & Vinciarelli, A. (2008). *Machine learning for audio, image and video analysis.* Springer. 1st. edition.

Cataltepe, Z., Yaslan, Y., & Sonmez, A. (2007). Music genre classification using midi and audio features. *EURASIP Journal on Advances in Signal Processing, 2007.*

Fujishima, T. (1999). Realtime chord recognition of musical sound: A system using common lisp music. *Proceedings of the International Computer Music Conference.* Beijing.

Gómez, E., & Herrera, P. (2004). Automatic extraction of tonal metadata from polyphonic audio recordings. *Proceedings of 25th International AES Conference.* London, UK.

Lidy, T., Rauber, A., Pertusa, A., & Iñesta, J. (2007). Improving genre classification by combination of audio and symbolic descriptors using a transcription system. *Proceedings of the ISMIR* (pp. 61–66). Vienna, Austria.

Piston, W. (1987). *Harmony.* Norton, W. W. & Company, Inc. 5th. edition.

Using Mathematical Morphology for Geometric Music Retrieval

Mikko Karvonen, Kjell Lemström {MTKARVON,KLEMSTRO}@CS.HELSINKI.FI
Department of Computer Science, University of Helsinki

Abstract

In this paper, we discuss the applicability of the mathematical morphology framework that was originally developed for image processing, to music retrieval purposes. More specifically, MM techniques are adopted for finding approximate occurrences of symbolically encoded musical query patterns in symbolic music databases. We consider four possibilities to accomplish the task.

1. Introduction

Mathematical morphology (MM) is a tool regularly used in binary and greyscale image processing. MM is often used for template matching and pattern recognition; it can be used for extracting attributes, geometrical information or ”meaning” in images.

In this paper, we show how to apply the framework for content-based music retrieval for symbolically encoded music. To this end, we represent symbolic music as 2-D binary images in the integer space $\mathbb{Z}^2$ (cf. the well-known piano-roll representation). The note objects, individual pixels (1-pixels) or line segments, are sets of pixels in $\mathbb{Z}^2$. 1-pixel objects can represent the musical pitch and rhythm (henceforth referred to as the note-on representation); line segments provide also for representing note durations (the line-segment representation; see Fig. 1). This way, an obvious method to find occurrences of an image of note objects (the query pattern) within another image (the database) would be to use classical linear correlation. However, in an approximative case with the note-on representation, where the pixels of the query are slightly jittered in the database image, there is no correlation when operating with binary images. We will show how to obtain robust matching tools to overcome this problem by adopting appropriate MM techniques.

1.1. Basic morphologic operations

The elementary binary MM operations, dilation and erosion, are based on the Minkowski addition and subtraction (Heijmans, 1995). They are nonlinear neighborhood operations on two sets. The typically smaller one is called structuring element (SE). Dilation performs a maximum on SE, which has a growing effect on the target set, while erosion performs a minimum on SE and causes the target set to shrink. Dilation can be used to fill gaps in an image, erosion, for example, for removing salt-and-pepper type noise.

A straight-forward template-matching operator, the hit-or-miss transformation (HMT), is based on these elementary operations. HMT is the intersection between an image eroded by SE_1 and the image's complement eroded by SE_2, where SE_2 is a local background of SE_1 (Bloomberg & Maragos, 1990). The output of the operation is an image where only the occurrences are marked with an ”on” pixel (see Fig. 1). HMT's weakness is that it finds only exact matches. In (Bloomberg & Maragos, 1990), two modified HMT approaches are presented to fight this problem.

2. Algorithms

We adopted four possibilities and implemented them in MATLAB. For the present, we do not have an automatic conversion tool. The musical examples were generated by loading MIDI songs into a sequencer and then taking binary-valued screenshots of the corresponding piano-roll illustrations.

2.1. Correlation & morphologic preprocessing

In the note-on representation, approximative matching is performed by replacing the 1-pixels in one image by filled discs (dilation). Thus, for instance, in a jittered query the pixels in the query image fall within the discs in the database. Intuitively, the disc allows for approximation both time- and pitchwise, and its diameter gives the magnitude of the approximation.

2.2. Threshold convolution

In order to use another HMT technique, based on threshold convolution (TC) (Bloomberg & Maragos, 1990), let us represent the database and query as sets

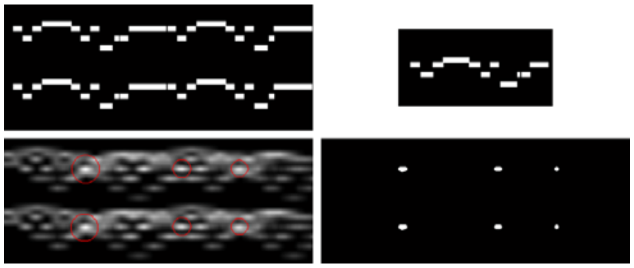

Figure 1. A) Piano-roll excerpt of Sibelius' Finlandia. B) Query pattern. C) TC between query and image. D) Occurrences found by applying Heaviside's USF on TC.

X and W, respectively, and define 2-D functions $x(n)$ and $w(n)$, such that $x(n) = 1$ if $n \in X$ and $x(n) = 0$ if $n \in X^c$ (accordingly for $w(n)$). We apply a threshold value θ to the convolution of x and w, and use Heaviside's unit step function to decide which convoluted pixels are to be used. If $\theta = |W|$, the operation is equivalent to the erosion of X by W. By lowering θ, we can impose looser conditions. It is notable that convolving a binary signal x with a binary template w and comparing the result to a threshold is actually the optimum decision for detecting w in x, when x contains a shifted version of w corrupted with binary salt-and-pepper noise (Bloomberg & Maragos, 1990).

2.3. Rank order filtering

Rank order filtering (ROF) is theoretically equivalent to TC, and it is optimal for template detection (Bloomberg & Maragos, 1990). In a morphological image-processing scenario, when the SE consists only of hits, ROF is shift-invariant and increasing (erosion and dilation are special cases). ROF operations are more robust against shape distortion than erosion, but they are computationally more complex.

A set-theoretic definition of a binary ROF, which avoids sorting and uses only pixel counting, was presented in (Maragos & Schafer, 1989). The definition gives the r-th rank order transformation ($r = 1, 2, \ldots, |W|$) of the binary set X by a binary W. In the transformation, W is shifted step-by-step. When located at point $z \in \mathbb{Z}^2$, we always select the r-th largest of the points belonging to the intersection $X \cap (W + z)$. If $r = |W|$, the rank operation becomes an erosion and the filter operates as HMT. By lowering the rank one can obtain better results. Note also that TC and ROF yield similar results when $\theta = r$.

2.4. Blur HMT

The blur hit-or-miss transformation (BHMT) is a simple extension to the classic HMT, providing tolerance to salt-and-pepper noise and image alignment errors. BHMT alters the image, but preserves the shape of SEs. The "blur" parameter specifies the maximum distance allowed for a match in between the query pixel and the corresponding pixel in the database image. The operator is implementable using Boolean image operators, which makes it faster than integer-based techniques, such as ROF(Bloomberg & Vincent, 2000).

BHMT requires that there is (1) an ON pixel within a radius of r_1 of each hit, and (2) an OFF pixel within a radius r_2 of each miss (Bloomberg & Maragos, 1990). For SEs that have a specific shape (e.g., a melody pattern), the blur match gives more immunity to pixel noise that occurs near the boundaries. As compared to the original HMT operation, the only change is the inclusion of a straight-forward preprocessing: the image is first dilated by a disc-shaped SE_1 and then the image's complement by another disc, SE_2.

3. Conclusions

We introduced 4 algorithms based on MM that are applicable to approximate content-based retrieval on symbolically encoded polyphonic music. Based on our preliminary experiments the approach seems promising: all the methods were capable of finding occurrences of query patterns that had minor distortions, such as local jittering. In note-on representation, with a small database, ROF was the fastest; with larger databases TC outperformed the others. However, we expect BHMT to be the choice when implemented by using Boolean image operations. With line segments, preprocessed correlation was the fastest but it also gave false positives most often. Further conclusions are left for the future when we have an automatic conversion tool and C implementations of the algorithms.

References

Bloomberg, D., & Maragos, P. (1990). Generalized hit-miss operators with applications to document image analysis. *SPIE Conference on Image Algebra and Morphological Image Processing* (pp. 116–128).

Bloomberg, D., & Vincent, L. (2000). Pattern matching using the blur hit-miss transform. *Journal of Electronic Imaging*, *9*, 140–150.

Heijmans, H. (1995). Mathematical morphology: A modern approach in image processing based on algebra and geometry. *SIAM Review*, *37*, 1–36.

Maragos, P., & Schafer, R. (1989). Morphological filters. part ii: Their relations to median, order-statistic, and stack filters. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, *35*, 1153–1184.

Learning to analyse tonal music

Plácido R. Illescas PLACIDOROMAN@GMAIL.COM
David Rizo DRIZO@DLSI.UA.ES
José Manuel Iñesta INESTA@DLSI.UA.ES
Departamento de Lenguajes y Sistemas Informticos, Universidad de Alicante, Spain

Keywords: harmony, tonal analysis, genetic algorithms

Abstract

This work is an effort towards the development of a system for the automation of traditional tonal analysis of polyphonic scores in symbolic format. The system detects chords with their tonal functions, and key changes. All the possible tonal and key analyses are represented as a weighted directed acyclic graph. The best analysis is the path that maximizes, through a dynamic programming algorithm, the sum of weights in the graph. The selection of the weights according to the importance of each possible harmonic progression is a key issue. A genetic algorithm is proposed to learn them from a training corpus of a given music style. The proof of concept of this approach has been tested on Bach chorales.

1. Introduction

Musical analysis is a means to better understand the thought of the composer when creating a piece. A musician must perform a good musical analysis to execute a correct interpretation of a work. The melodic, harmonic, and tonal function analyses are the basic elements in order to achieve an optimal musical analysis. Besides, there are many applications of automatic analysis to diverse areas of music: education, score reduction, pitch spelling, harmonic comparison of works, etc.

The automatic tonal analysis has been tackled under different approaches and objectives. Some works use grammars to solve the problem (Winograd, 1992), or probabilistic models like that in (Raphael & Stoddard, 2004) and others based on preference rules or scoring techniques (Temperley & Sleator, 1999). The *music theory workbenck* (MTW) system (Taube, 1999) solves the problem by means of model matching. A more comprehensive review of these works can be found in (Barthelemy, 2001). To the best of our knowledge, there is no work nowadays that learns from a training corpus.

Our objective is not only to obtain a high percentage of correct analyses, but also to describe in a human-readable way the reasons why the system has chosen an analysis.

2. Methodology

In order to analyze a musical piece the system segments each bar into a number of time windows, all possible chords are obtained from the notes in each. After this, the valid keys for each window are selected, given the accidentals of the notes involved. From these data, a weighted acyclic directed graph (wDAG) organized by layers is built. Each layer represents a window. The nodes of the graph correspond to chords with tonal functions in a tonality. The edges of the graph represent the valid progressions between the nodes in successive layers, weighted according to the importance of those progressions in order to establish a tonality.

Once the graph is built, a dynamic-programming approach is utilized to compute the best path along the graph, discovering the best tonality and tonal function sequence. The output is the Roman numeral analysis with tonality segmentation.

The main problem here is how to establish the values for those weights, because the performance of the system is very sensitive to them. Moreover, these values may be conditioned by the music genre. For example, valid progressions for jazz music were prohibited in Baroque. Therefore, a method able to adjust the weight values from a training set of a given target musical genre is desirable.

Our proposal is to use a genetic algorithm to do this task. The chromosome encodes the set of weights to be optimized. Each gene represents a weight. The range values valid for each weight are based on those used in (Illescas et al., 2007).

The fitness function evaluates their suitability for a correct analysis. A set of Bach chorales have been analyzed by an expert. The algorithm tries to minimize the number of errors made by the analysis system when compared with the expert's in a window frame basis. This fitness function has proven to render parameter values close to those that would be empirical set by an expert for analyzing Bach chorales.

The system has been implemented using the JGAP [1] package using the default configuration.

3. Experiments

To test the system, transcriptions in MusicXML format of the harmonized chorals from J.S.Bach (BWV-253, 26, 437, 29, 272, and 438) have been used. The manually tagged corpus can be downloaded from our website quoted below. (http://grfia.dlsi.ua.es/cm).

Table 1. Compared tonality (T.) and tonal function (T.F.) success rates for the system using fixed weights and the system with weights learnt by the genetic algorithm

	Without GA		With GA	
	T.F.	T.	T.F.	T.
BWV-26	80	64	86	73
BWV-272	61	21	66	50
BWV-29	79	50	77	50
BWV-253	79	15	64	20
BWV-437	71	56	70	72
BWV-438	72	71	61	86

The weights are learnt leaving one choral out for the testing. 872 time windows have been analyzed. The results in Table 1 show the genetic algorithm approach outperforms the system using weights established by a human expert. Most of the errors occur in the tonality modes.

Since two different analyses sometimes can be both valid we cannot give success percentages in order to compare to those reported by MTW. From our point of view, the MTW fails in some tonal function progressions and seems to make mistakes when analyzing alternative tonalities by not solving the chord cadence (e.g. BWV 2-6 at bar 3, beats 2-4). Our system corrects those errors by means of the cadence scoring. However, we must correct some errors the MTW does not make.

[1] http://jgap.sourceforge.net/

4. Discussion and conclusions

This paper presents a system to analyze automatically a score from the melodic and harmonic points of view, providing the tonality changes and the Roman numeral analysis of each chord along with its tonal function.

The system performs comparably to MTW, but it has the advantage to be ready to work with monodic melodies only adding more possibilities of analysis at each layer of the graph.

The use of the learning system avoids the arbitrariness or subjectivity of a set of values given by a human expert and, in addition, it permits the system to fit its performance to different music genres.

Acknowedgements

This work is supported by the Spanish PROSEMUS project (TIN2006-14932-C02) an the research programme Consolider Ingenio 2010 (MIPRCV, CSD2007-00018).

References

Barthelemy, J. (2001). Figured bass and tonality recognition. *ISMIR*.

Illescas, P. R., Rizo, D., & Iñesta, J. M. (2007). Harmonic, melodic, and functional automatic analysis. *Proceedings of the 2007 International Computer Music Conference* (pp. 165–168).

Raphael, C., & Stoddard, J. (2004). Functional harmonic analysis using probabilistic models. *Comput. Music J.*, *28*, 45–52.

Taube, H. (1999). Automatic tonal analysis: Toward the implementation of a music theory workbench. *Comput. Music J.*, *23*, 18–32.

Temperley, D., & Sleator, D. (1999). Modeling meter and harmony: A preference-rule approach. *Comput. Music J.*, *23*, 10–27.

Winograd, T. (1992). *Linguistics and the computer analysis of tonal harmony*. Cambridge, MA, USA: MIT Press.

Chorale Harmonization in the Style of J.S. Bach
A Machine Learning Approach

Alex Chilvers ACHILVER@ICS.MQ.EDU.AU
Macquarie University, NSW 2109 North Ryde, Australia

Menno van Zaanen MVZAANEN@UVT.NL
Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Abstract

In this article, we present a new approach to the task of automatic chorale harmonization. Taking symbolic musical transcripts as input, we use the TiMBL Machine Learning package to generate harmonizations of unseen chorale melodies. Extracting features from the soprano notes only, the system generates the other three melody lines in the chorale. We experiment with features denoting different properties of notes, such as pitch, note length, location in the bar or piece, and features of surrounding notes, but also with global features such as metre and key. The current system harmonizes 41.71% of the data exactly like J.S. Bach did (modulo octave).

1. Introduction

Automatic harmonization, where accompanying melody lines are generated given a single melody line, is one of the oldest topics to be explored by computer musicians. A relatively early example of rule-based automatic harmonization using a constraint-based system is described in Pachet and Roy (2001). Other approaches such as the use of probabilistic finite state grammars (Conklin & Witten, 1995) or neural networks (Hild et al., 1992) have also been explored. Obviously, there is much more literature, but space restrictions do not allow us to provide more.

In this article we will use chorales composed by Johann Sebastian Bach. These chorales have long been the popular choice as data for computer music researchers exploring the harmonization challenge. The simple fact that there are many of these relatively short pieces, all composed in Bach's recognizable style, makes them appealing for this task.

Furthermore, it is known that Bach began with another composer's melody. This melody is used as the soprano voice in the chorales. He then harmonized this melody by adding notes for the alto, tenor, and bass voices, effectively adding his own accompaniment.

In the machine learning (ML) task described here, the same approach is used. Starting from properties extracted from the original melody, the soprano line, we apply a ML classifier, which outputs notes for the other three melody lines.

2. Approach

We treat the task of automatic harmonization as a classification task, which means that we can use standard ML techniques. As input, notes from the soprano melody are used. The output is a chord (i.e. notes for the three additional melody lines) describing the harmonization.

There are many other ways to approach the task. For example, one could classify each voice separately. However, initial experiments using that approach produced discouraging results.

2.1. Classifications

The output of the system is encoded using 12-bit strings, where a single bit represents whether or not the corresponding note is present in the chord. The output is represented relative to the chorale's tonic to avoid data sparseness problems. This means that the first bit in the bit string corresponds to the tonic, the second bit denotes a semi-tone higher than the tonic, and so on. For example, a major triad is represented by the string `100010010000`. Due to the three-voice nature of chorale harmonization, each bit string can contain a maximum of three `1`s.

Using this encoding, the information on the exact octave of a note is lost. Also, the ordering of voices is not explicitly encoded, as all notes are put in a bit vector which orders notes with respect to the tonic.

2.2. Feature vectors

The classifier takes properties from each of the initial (soprano) notes as input. These properties are encoded as features in feature vectors. The choice of features has a major impact on the performance of the system, so we consider a number of different approaches and combinations.

Local features encode information of the soprano notes. We use pitch and duration. Pitch is represented using the semi-tonal distance from the tonic. For example, when the note is the tonic, the value is `0`, whereas a whole tone above the tonic has value `2`.

Contextual features represent information of surrounding notes. We take the context to be the previous and next `n` soprano notes. This `n` is the size of the context and can be varied in different experiments. In the experiments described here, `n` is set to three.

With the goal of producing a nicely flowing harmonization, we have experimented with the previous `m` classifications, i.e. generated chords, as features. Previous attempts at automatic harmonization have been made using Markov models (Biyikoglu, 2003) and probabilistic inference (Allan & Williams, 2005), which suggest that using harmonization of previous notes can help to predict chords. Initial experiments showed that previous classifications have a negative impact. Many of them are wrong, which means that many incorrect values are incorporated into the feature vectors.

Other contextual features that are used are the location of a note in a bar and the location of a bar in a piece.

Finally, we incorporated global features, that provide information on the entire piece. In these experiments, we have used whether the piece is major or minor, and the piece's metre.

3. Results

For the experiments, we took 230 J.S. Bach chorales. The performance of the system was measured by calculating accuracy of the classified chords.[1] The results were computed using 10-fold cross validation.

As a baseline, the majority class, which is the major triad, is used. This results in an accuracy of 8.71 with a standard deviation of 0.86.

[1]We only count exactly matching chords and do not use other musically correct chords, to reduce subjectivity.

For the ML approach, we used the TiMBL package (Daelemans et al., 2002), which provides optimized k-NN classification algorithms. However, other classifiers can be used as well.

The best TiMBL system produced an accuracy of 41.71 with a standard deviation of 5.80. This result was acquired using all of the previously mentioned features, except for the previous classification features.

4. Conclusion

In this article we showed that automatic harmonization can be treated as a ML classification task. The results, which are generated using pitch, duration, key, positional, and contextual features, greatly outperform the majority class baseline. However, the current experiments simplify the task by removing octave and voice information.

There are many extensions that may improve the current system. Firstly, since the best results are generated using (almost) all features, it may very well be the case that additional features will further improve results. Secondly, the accuracy measure only takes into account harmonization exactly as Bach composed it. The way accuracy is measured now, chords counted as incorrect may actually still be musically valid.

References

Allan, M., & Williams, C. K. I. (2005). Harmonising chorales by probabilistic inference. *Advances in Neural Information Processing Systems 17.* MIT Press.

Biyikoglu, K. M. (2003). A markov model for chorale harmonization. *Proceedings of the 5th Triennial ESCOM Conference.*

Conklin, D., & Witten, I. H. (1995). Multiple viewpoint systems for musical prediction. *Journal of New Music Research*, *24*, 51–73.

Daelemans, W., Zavrel, J., van der Sloot, K., & van den Bosch, A. (2002). *TiMBL: Tilburg memory-based learner* (Technical Report ILK 02-10). Tilburg University, Tilburg, the Netherlands.

Hild, H., Feulner, J., & Menzel, W. (1992). HARMONET: A neural net for harmonizing chorales in the style of J.S. Bach. *Advances in Neural Information Processing 4 (NIPS 4).*

Pachet, F., & Roy, P. (2001). Musical harmonization with constraints: A survey. *Constraints*, *6(1)*, 7–19.

An information-dynamic model of melodic segmentation

Marcus T. Pearce M.PEARCE@GOLD.AC.UK
Daniel Müllensiefen D.MULLENSIEFEN@GOLD.AC.UK
Geraint A. Wiggins G.WIGGINS@GOLD.AC.UK
Centre for Cognition, Computation and Culture
Goldsmiths, University of London
New Cross, London SE14 6NW, UK

Keywords: segmentation, grouping, melody, unsupervised learning, information theory

Abstract

We present a new model of melodic segmentation based on unsupervised information-dynamic analysis. This is justified theoretically on the basis of previous research in musicology, computational linguistics, machine learning and experimental psychology. The model is evaluated in comparison with existing rule-based algorithms on a melodic boundary detection task. Although the model is outperformed, it does contribute to a Hybrid model that achieves superior performance, which is surprising given that it uses unsupervised learning and has not been optimised for the task of melodic segmentation.

Melodic segmentation is a fundamental process in the perception, composition, performance and analysis of music whilst also being necessary for melody indexing and retrieval in MIR applications. Most existing models of melodic segmentation use rules based on music-theoretic knowledge. As a result, they are style-specific or require expert knowledge of the musical style to tune their parameters. Our goal here is to develop a more general model of melodic segmentation based on unsupervised machine learning.

The *IDyOM* (Information Dynamics Of Music) model is based on music-theoretic proposals that perceptual groups are associated with points of closure where the ongoing cognitive process of expectation is disrupted because the context fails to stimulate strong expectations for any continuation or because the actual continuation is unexpected (Meyer, 1957; Narmour, 1990). We give these proposals precise definitions in an information-theoretic framework in which *information content* represents the unexpectedness of an outcome and *entropy* the uncertainty of a prediction. We propose that boundaries will occur before events for which unexpectedness and uncertainty are high.

Further support comes from empirical psychological research demonstrating that infants and adults use implicitly learnt statistical properties of pitch sequences to find segment boundaries on the basis of higher transition probabilities within than between groups (Saffran et al., 1999). In machine learning and computational linguistics, algorithms based on the idea of segmenting before events associated with high uncertainty or surprise perform reasonably well in identifying word boundaries in infant-directed speech (Brent, 1999; Cohen et al., 2007).

Our information-theoretic measures are derived by estimating the conditional probabilities of sequential melodic events using a sophisticated statistical model of melodic structure (Conklin & Witten, 1995; Pearce, 2005) that accounts well for human melodic expectations (Pearce, 2005). This model is able to combine n-gram statistics of several orders, use a variable global order bound, predict multiple attributes of a melodic event, use statistical regularities in different derived representations and combine schematic models based on long-term musical exposure with short-term models based on the local properties of the music currently being analysed. In this work, we use the model to predict the pitch, IOI (inter-onset interval) and OOI (offset-to-onset interval) associated with melodic events, multiplying the probabilities of these attributes together to yield the event's probability. We focus on the unexpectedness of events (information content) using this as a boundary strength profile, picking local peaks to identify boundary locations. The role of entropy will be considered in future work.

We test the model on a boundary identification task

Model	F1	Precision	Recall
Hybrid	0.66	0.87	0.56
Grouper	0.66	0.71	0.62
LBDM	0.63	0.70	0.60
GPR2a	0.58	0.99	0.45
IDyOM	0.58	0.76	0.50
GPR2b	0.39	0.47	0.42
GPR3a	0.35	0.29	0.46
GPR3d	0.31	0.66	0.22
Always	0.22	0.13	1.00
Never	0.00	0.00	0.00

Table 1. Mean performance over 1705 melodies.

in the largest database of German folk songs (Erk) in the Essen collection (Schaffrath, 1995) containing 1705 melodies annotated with phrase boundaries by a musicologist. 10-fold cross-validation is used to train the model and obtain information-content profiles for each unseen melody in the dataset. Performance is compared with several existing rule-based models: LBDM (Cambouropoulos, 2001), Grouper (Temperley, 2001), the Grouping Preference Rules 2a, 2b, 3a, 3d (Lerdahl & Jackendoff, 1983; Frankland & Cohen, 2004) and two baseline models (always, never). We use *F1*, *Precision* and *Recall* to assess the correspondence between the models' boundaries and the target boundaries.

Mean performance measures are shown in Table 1. The four models achieving mean F1 values of over 0.5 (Grouper, LBDM, GPR2a and IDyOM) were chosen for further analysis. Sign tests between the F1 scores on each melody indicate that all differences between these models are significant at an alpha level of 0.01, with the exception of GPR2a and LBDM. The four models were entered as predictor variables in a logistic regression analysis. Stepwise backwards elimination using the Bayes Information Criterion failed to remove any of these predictors and the resulting Hybrid model (which includes IDyOM) produced an F1 performance that was significantly better than Grouper.

IDyOM contains no hard-coded music-theoretic rules but learns regularities in the melodic data it is trained on and outputs probabilities of note events which are ultimately used to derive information content for each note event in a melody. It therefore might be expected to outperform rule-based models on more unfamiliar musical territory (e.g., non-Western music). Furthermore, as a model of prediction and expectation in music, it has not been specifically optimised for melodic segmentations. While future optimisation (e.g., in representation) might be expected to yield improvements in segmentation performance, the present results demonstrate a link between sequential predictive information content and melodic segmentation.

References

Brent, M. R. (1999). An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, *34*, 71–105.

Cambouropoulos, E. (2001). The local boundary detection model (LBDM) and its application in the study of expressive timing. *Proceedings of the International Computer Music Conference* (pp. 17–22). San Francisco: ICMA.

Cohen, P. R., Adams, N., & Heeringa, B. (2007). Voting experts: An unsupervised algorithm for segmenting sequences. *Intelligent Data Analysis*, *11*, 607–625.

Conklin, D., & Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, *24*, 51–73.

Frankland, B. W., & Cohen, A. J. (2004). Parsing of melody: Quantification and testing of the local grouping rules of Lerdahl and Jackendoff's *A Generative Theory of Tonal Music*. *Music Perception*, *21*, 499–543.

Lerdahl, F., & Jackendoff, R. (1983). *A generative theory of tonal music*. Cambridge, MA: MIT Press.

Meyer, L. B. (1957). Meaning in music and information theory. *Journal of Aesthetics and Art Criticism*, *15*, 412–424.

Narmour, E. (1990). *The analysis and cognition of basic melodic structures: The implication-realisation model*. Chicago: University of Chicago Press.

Pearce, M. T. (2005). *The construction and evaluation of statistical models of melodic structure in music perception and composition*. Doctoral dissertation, Department of Computing, City University, London, UK.

Saffran, J. R., Johnson, E. K., Aslin, R. N., & Newport, E. L. (1999). Statistical learning of tone sequences by human infants and adults. *Cognition*, *70*, 27–52.

Schaffrath, H. (1995). The Essen folksong collection. In D. Huron (Ed.), *Database containing 6,255 folksong transcriptions in the Kern format and a 34-page research guide [computer database]*. Menlo Park, CA: CCARH.

Temperley, D. (2001). *The cognition of basic musical structures*. Cambridge, MA: MIT Press.

Discovery of distinctive patterns in music

Darrell Conklin CONKLIN@CITY.AC.UK
Department of Computing, City University London, UK

Keywords: pattern discovery, distinctive pattern, anticorpus, folk songs

Abstract

This paper proposes a new view of pattern discovery in music: mining of a corpus for maximally general patterns that are overrepresented with respect to an anticorpus. An algorithm for maximally general distinctive pattern discovery is described and applied to folk song melodies from three geographic regions. Distinctive patterns have applicability to comparative music analysis tasks where an anticorpus can be defined and contrasted with an analysis corpus.

1. Introduction

Pattern discovery in music is a central part of symbolic music processing, with a wide range of applications, including motivic analysis, music generation, and music classification.

Pattern discovery in music has typically been viewed as an unsupervised learning problem, with the attendant difficulty of ranking discovered patterns. This difficulty arises because frequency of occurrence and specificity are inversely related, and any ranking measure must balance these two quantities. Statistical approaches for pattern ranking (Conklin & Anagnostopoulou, 2001) achieve this using a *background model*, for example a zero or first order Markov model. The background model is used to compute expected counts, which are compared to observed pattern counts for large deviations.

The use of a background model, possibly estimated from the analysis corpus itself, raises concerns regarding how “musical” are pieces implicitly generated by the background model. Therefore, the significance of patterns may easily be overestimated, leading to the presentation of many irrelevant patterns.

By analogy with work in bioinformatics on discriminative motif discovery (Barash et al., 2001; Redhead & Bailey, 2007), explicit negative pieces may provide a better alternative to using a background model.

Adopting this view for music, patterns can be sought that are useful for classification: able to discriminate between a corpus and an anticorpus (the division is left to the analyst: whether by style, composer, period, geographic region, etc.). Though patterns have been used previously for supervised learning tasks, most approaches rely on an exhaustive catalog of fixed-width patterns.

This paper introduces the *distinctive pattern* for music and develops an algorithm for their discovery in a corpus. As a proof of concept, the method is illustrated on a few sections from the Essen folk song database to find *maximally general* distinctive patterns: those that are least likely to overfit the corpus and therefore most likely to be useful for classification.

2. Distinctive patterns

A *distinctive pattern* is one that is overrepresented in a corpus as compared to an anticorpus. A likelihood ratio is used to evaluate the distinctiveness or interest of a pattern (see Glossary of Notation in Table 1).

Table 1. Glossary of Notation.

Term	Meaning
$c^{\oplus}(P)$	count of pattern P in the corpus
$c^{\ominus}(P)$	count of pattern P in the anticorpus
$n^{\oplus}$	size of the corpus
$n^{\ominus}$	size of the anticorpus
$I(P)$	likelihood ratio of pattern P

To define the likelihood ratio, both pattern P and class ($\oplus$: corpus or $\ominus$: anticorpus) can be viewed as indicator variables of pieces (a pattern indicated if it occurs in a piece, and the class indicated from by corpus and anticorpus labelling). The likelihood ratio measures the relative probability of the pattern in the corpus

Table 2. Maximally General Distinctive Patterns in Three Folk Song Regions.

Corpus	P	$p(P\|\oplus)$	$p(P\|\ominus)$	$I(P)$
Shanxi	`[{contour:-,diaintc:M6},{contour:+,diaintc:m7}]`	30/237	0/195	inf
Austrian	`[{rest:0,diaintc:m2},{diaintc:M3}]`	34/102	14/330	7.9
Swiss	`[{diaintc:M6},{diaintc:M2}]`	22/93	12/339	6.7

and anticorpus, or equivalently, its overrepresentation measured by the ratio of its observed count in the corpus to its expected count given the anticorpus probability:

$$I(P) \overset{\text{def}}{=} \frac{p(P|\oplus)}{p(P|\ominus)} = \frac{c^{\oplus}(P)}{p(P|\ominus) \times n^{\oplus}}$$

The algorithm used to search for distinctive patterns is a sequential pattern mining method using a feature set (Conklin & Bergeron, 2008) pattern representation. A depth-first search of a specialization space uses two refinement operators at a search node: addition of a feature to the last set of a pattern, and extension of the pattern by the empty feature set. The search is lexicographic and therefore no search node is visited more than once. Given that *maximally general* distinctive patterns are sought, a branch in the search tree may be immediately terminated with success if a minimal specified pattern interest is achieved by the pattern at a search node.

3. Results

Three geographic regions represented in the Essen folk song database were chosen to illustrate the method (Shanxi: 237 pieces; Austrian: 102; Swiss: 93). Austrian and Swiss were chosen as a pair due to their geographic proximity. Patterns were considered interesting if $p(P|\oplus) \geq 0.2$ (in at least 20% of the corpus) and $I(P) \geq 3$ (at least 3-fold overrepresented). An extensive catalog of abstract musical viewpoints (functions that compute features from events) was provided to the algorithm.

Many maximally general distinctive patterns are found; Table 2 indicates, for each region, one discovered pattern with high interest. The presented pattern is therefore highly distinctive when the other two regions are taken together as the anticorpus.

In the table, several viewpoints appear in patterns: those referencing diatonic interval class, melodic contour, and a viewpoint indicating whether the event follows a rest. The patterns are very general and in some cases quite surprising (e.g., for Swiss: simply a diatonic interval class of a major sixth followed by a major second) and cover many pieces. The Shanxi pattern is maximally distinctive: not covering any pieces in the anticorpus.

4. Discussion

This paper has defined distinctive pattern discovery and illustrated an application using a few sections of the Essen folk song database. Distinctive patterns can have musicological interest in isolation, or can be viewed as binary global features for describing pieces for supervised learning. In this sense, distinctive pattern discovery in music can be viewed as a feature selection task for classification.

The pattern discovery algorithm has also been used for motivic analysis of a single piece (Conklin, 2008), showing that distinctive patterns corresponding to known motives are discovered. It is also under development for bioinformatics motif discovery applications.

References

Barash, Y., Bejerano, G., & Friedman, N. (2001). A simple hyper-geometric approach for discovering putative transcription factor binding sites. *WABI 2001: First International Workshop on Algorithms in Bioinformatics* (p. 278). Springer.

Conklin, D. (2008). Mining for distinctive patterns in the first movement of Brahms's String Quartet in C Minor. *MaMuX International Workshop on Computational Music Analysis.* IRCAM, Paris, April 5. Abstract.

Conklin, D., & Anagnostopoulou, C. (2001). Representation and discovery of multiple viewpoint patterns. *Proceedings of the International Computer Music Conference* (pp. 479–485). Havana.

Conklin, D., & Bergeron, M. (2008). Feature set patterns in music. *Computer Music Journal*, *32*, 60–70.

Redhead, E., & Bailey, T. (2007). Discriminative motif discovery in DNA and protein sequences using the DEME algorithm. *BMC Bioinformatics*, *8*, 335.

Melody Characterization by a Fuzzy Rule System

Pedro J. Ponce de León, David Rizo {PIERRE,DRIZO}@DLSI.UA.ES
Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Spain

Rafael Ramirez RRAMIREZ@IUA.UPF.EDU
Music Technology Group, Universitat Pompeu-Fabra, Barcelona, Spain

Abstract

We present preliminary work on automatic human-readable melody characterization. In order to obtain such characterization, we (1) extract a set of statistical descriptors from a dataset of MIDI files, (2) apply a rule induction algorithm to obtain a set of (crisp) classification rules for melody track identification, and (3) automatically transform the crisp rules into fuzzy rules by applying a genetic algorithm to generate the membership functions for the rule attributes. We present and discuss some results.

1. Melody characterization

Melody is a somewhat elusive musical term that often refers to a central part of a music piece that catches most of the listener's attention, and which the rest of music parts are subordinated to. This is one of many definitions that can be found in many places, particularly music theory manuals. However, these are all formal but subjective definitions given by humans. The goal in this work is to automatically obtain an objective and human friendly characterization of what it is considered to be a melody.

The identification of a melody track is relevant for a number of applications like melody matching (Uitdenbogerd & Zobel, 1999), motif extraction from score databases, or extracting melodic ringtones from MIDI files. In this work we approach the problem of automatically finding a melody track in a MIDI file by using statistical properties of the musical content. The melody model is a set of human-readable fuzzy rules (Cordón & Herrera, 1995), automatically induced from a corpora of MIDI files.

The rest of the paper is organized as follows: first, the statistical MIDI track content description is outlined. Then, a first (crisp) rule set for melody characterization is induced. Next, we describe how we fuzzify the crisp set of rules in order to improve rule readability. Finally some results and conclusions are discussed.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. MIDI track content description

MIDI track content is described by a collection of statistics on several properties of musical note streams, such as pitch, pitch interval and note duration, as well as track properties such as number of notes in the track, track duration, polyphony rate and occupation rate. As a result, MIDI tracks are represented by real number vectors with 34 statistical values. In the rule system presented in section 3 only 13 of these attributes are used. Each track has been both manually and automatically labeled as being a melody track or a non-melody track. Details on such statistics and the labeling process can be found in (Ponce de León et al., 2007).

3. A rule system for melody characterization

In previous works, several decision tree and rule classification algorithms have been used to classify MIDI tracks as melody or non-melody. In this work, a rule system obtained using the RIPPER algorithm (Cohen, 1995) is used as the basis to induce a fuzzy rule system. An example of an induced (crisp) rule is as follows:

```
if (AvgPitch >= 65.0) and (OccupRate >= 0.51)
and (AvgInterval <= 3.64) and (NumNotes >= 253)
then IsMelody=true
```

4. From crisp to fuzzy rule system

A fuzzyfication process is applied to such crisp rule system, in order to obtain a user friendly rule system.

Two basic steps must be carried out for the fuzzyfication of the crisp rule system. First, the data representation must be fuzzified. That is, numerical input and output values must be converted to fuzzy terms. Second, the rules themselves must be translated into fuzzy rules, substituting linguistic terms for numerical boundaries.

4.1. Fuzzyfying attributes

Fuzzyfication of numerical attributes usually involves the participation of an human expert who provides domain knowledge for every attribute. Our approach in this paper is to replace the human expert by a genetic algorithm (GA) which automatically learns the fuzzy sets (Cordón & Herrera, 1995).

Prior to applying the GA, linguistic terms for every attribute are defined. For efficiency reasons, the shape for a fuzzy set in this work is restricted to be either trapezoidal or triangular, and the number of fuzzy sets per attribute is restricted from 3 to 5. A chromosome in the GA encodes the fuzzy sets for a fuzzy attribute, and an individual in the GA population represents all the fuzzy attributes.

The fitness function for the GA consists of testing each individual with a fuzzy inference system using the fuzzy rule system discussed in section 4.2 on a reference dataset (see section 5). The better the performance of the rule system, the better the individual's score. This is possible because rule fuzzification is a process independent from fuzzy set definition.

4.2. Crisp rule system fuzzyfication

The final step in this method is to fuzzify the rule system. Antecedents of the form ($x > v$) are translated into one or more antecedents of the form ($x\ IS\ T$), where T is a linguistic term defined for attribute x. The value v partitions the attribute domain in two subsets, and the direction of the inequality guides the selection of the fuzzy terms to be included in fuzzy antecedents.

In the present work, the crisp rule system (section 3) has been fuzziyfied in order to present a proof of concept of the methodology applied. A disjunctive fuzzy rule set is then obtained. Below is a fuzzy rule corresponding to the rule shown in section 3:

IF (AvgPitch IS *high* OR AvgPitch IS *veryHigh*)
AND (OccupRate IS NOT *void*) AND (OccupRate IS NOT *low*)
AND (AvgInterval IS NOT *fifth_or_higher*)
AND (AvgInterval IS NOT *fourth*) AND (NumNotes IS *high*)
THEN IsMelody IS *true*

5. Experiments

All datasets used to test the fuzzy rule system consist of MIDI files, where the melody tracks are tagged with a special string in their track name. A reference dataset of 600 songs (called SMALL in Table 1) has been used to obtain the crisp rule system. The rest of datasets are used for testing the system: RWC-G (Goto et al., 2003) (100 songs) and AJP (762 songs) are both multi-genre datasets of academic, popular, rock and jazz music, among more than ten genres.

Table 1 presents results from applying both the crisp and the fuzzy rule system to the test datasets. A track is classified as a melody track if it fires at least one rule [1]. Otherwise, the track is classified as non-melody. Precision, recall and F-measure are computed for the class 'IsMelody'.

Dataset	Precision	Recall	F	Error rate
SMALL (*crisp*)	0.89	0.87	0.88	0.05
SMALL (*fuzzy*)	0.57	0.97	0.72	0.15
RWC-G (*crisp*)	0.54	0.77	0.64	0.13
RWC-G (*fuzzy*)	0.31	0.80	0.45	0.28
AJP (*crisp*)	0.88	0.89	0.88	0.05
AJP (*fuzzy*)	0.55	0.96	0.70	0.17

Table 1. Melody track classification results.

As the results show, the fuzzyfied rule system performs poorly than the original crisp rule system. However, the recall is consistently better for the fuzzy classifier. It follows that most errors are false positives, that is, some non-melody tracks are classified as melody tracks.

6. Conclusions and further work

A fuzzy rule system for melody track characterization in MIDI files has been outlined. The somewhat poor performance of the fuzzy rule system could be improved by rising the probability threshold for firing a fuzzy rule. Also, enforcing more than one fuzzy rule to be fired could help improve the results. A smarter way of fuzzifying rules (using information theory measures) could be applied. As some attributes appear to be more frequently used than other in rule systems, weighting fuzzy sets proportionally to their relative importance in the rules could be a way for improvement.

Acknowledgments

Supported by the projects: GV06/166 and CICyT TIN2006–14932–C02, partially supported by EU ERDF.

References

Cohen, W. W. (1995). Fast effective rule induction. *Machine Learning: Proc.of the 12th Intl. Conf.*.

Cordón, O., & Herrera, F. (1995). A general study on genetic fuzzy systems. In J. Smith (Ed.), *Genetic algorithms in engineering and computer science*, chapter 3, 33–57.

Goto, M., et al. (2003). Rwc music database: Music genre database and musical instrument sound database. *Proc. of the 4th Intl. Conf. on Music Information Retrieval (ISMIR)*

Ponce de León, P. J., Rizo, D., & Iñesta, J. M. (2007). Towards a human-friendly melody characterization by automatically induced rules. *Proc. of the 8th ISMIR* (pp. 437–440).

Uitdenbogerd, A., & Zobel, J. (1999). Melodic matching techniques for large music databases. *Proc. of the 7th ACM Intl. Multimedia Conf. (Part 1)* (pp. 57–66).

[1] In the fuzzy rule system, the rule must be fired with probability greater than 0.5.

Detecting Changes in Musical Texture

Atte Tenkanen Department of Musicology, University of Turku - Finland, attenka@utu.fi
Fernando Gualda Sonic Arts Research Centre, Queen's University of Belfast - UK

Keywords: music analysis, pitch-class set theory, circle of fifths, comparison set analysis, fuzzy k-nn

Abstract

This paper presents an analytical method for exploring changes in musical texture, and thus, for finding the potential boundaries of sections in a musical piece. It also presents a new function which measures a pitch-class based relation between two pitch-class sets. The method can be described as follows: A feature vector is calculated for each note in a composition; each feature vector expresses the changes between the textures before and after the note associated with it; among all feature vectors, an analyst can select the most significant ones which will act as class members in a fuzzy k-nn classifier; all the vectors derived from a score are thereafter input to the classifier. Then the k-nn classifier generates a curve graph which represents an overview of textural changes in the piece.

1. Introduction

The music analytical method presented here is an extension of the so called *comparison set analysis* (CSA) by Huovinen & Tenkanen (2008). In the CSA musical units are segmented automatically into overlapping sets of same cardinality. These segments are then compared with a similar type of *comparison set* using some similarity or distance measure. The comparison set embodies a chosen musical property whose prevalence is evaluated in a composition. The result can be presented in the form of a time series plot.

Several musical features are utilized in the present paper. However, instead of selecting a comparison set for each measurable feature separately, an analyst chooses by hand some notes which are associated with the interesting musical characters, here, changes in the musical texture. These mark boundaries that may distinguish and define musical events articulating the musical form. In this case the 'comparison set' consists of selected feature vectors inputted as class members to a *fuzzy k-nn classifier* (Keller et al. 1985). Thus, the classifier frees an analyst from defining the exact attributes for the group of comparison sets by learning from the selected musical context, and *forms* a multidimensional member ensemble which works analogously to the 'comparison set'. Information related to all notes is used thereafter for classifying all multidimensional moments in a piece. The method is applied to the third movement of Symphony No. 5 by Jean Sibelius.

2. Feature extraction

Harmonic, rhythmic and melodic data is preprocessed by segmenting the note information of the *event list* using imbricated segmentation for each feature separately. Two consecutive time windows of equal size are moved over the musical piece.

For simplicity, we consider only note onsets to produce overlapping segments of set-classes and pitch-class sets. Vector types and relation functions used in distance measurements are presented in Table 1. REL similarity measure originates to David Lewin (1980). Overlapping rhythm sets are constructed from proportions between unique consecutive note onsets. In the case of the first three vector types, all vectors associated with the first window are compared with all the vectors in the other window. Averaged distances between these windows, regarding to each feature, are calculated and stored as *feature vectors* associated to each 'borderline' note in the piece. A new Circle-Of-Fifths (COF) based measure called *cofrel* (COF relation) is used to evaluate relations between untransposed pitch-class sets. Instrumental melodic movements act as a source for normalized transition probability matrices. 'Onset density ratio' is calculated using the number of unique onset ticks inside both windows. All the features were normalized to zero mean and unit variance.

Table 1. Vector types, relation functions and segmentation cardinalities used in the analysis.

Feature vectors	Cardinality	Distance/Similarity Relation measure
set-classes	4	REL
pitch-class sets	same segmentation	cofrel
rhythm sets	4	cosine distance
melodic transition vectors	4	euclidean dist. between transition probability matrices
note duration distributions	-	euclidean dist. between normalized distributions
onset density ratio	-	ratio: $\min(dens_1/dens_2, dens_2/dens_1)$, (≤ 1)

2.1. *COF-relation* -algorithm

To consider the relation between untransposed pitch-class sets a new algorithm was developed. By using the generator $\langle 7 \rangle$ of a cyclic group Z_{12} under addition modulo 12 all pitch classes can be obtained from a given pitch class. This property allows distance measurements between any two pitch classes. The COF-distance between pc-sets a fifth apart is defined as 1, it follows that the most distant pc-sets (e.g. C and F#) would have COF-distance of 6. To calculate the COF-based distance between any two pitch-class sets, it is intuitively acceptable and equitable to take all $n * m$ -pairs of COF-distances between the sets into account, where m and n are the cardinalities of the pitch-class sets. The *COF-relation* between two pitch-class sets is: $cofrel(A, B) = \sqrt{\sum_{i \in A, j \in B} (c_{ij})^2} / \sqrt{(|A| * |B|)}$ where c_{ij} includes all the COF-distances between the pitch-class sets A and B.

3. Analysis example and some results

Eleven feature vectors connected to the lowest notes in the beginning of bars 105, 165, 213, 221, 242, 371, 407, 421, 427, 445 and 467 were selected as class 1 member vectors: There are remarkable changes in the texture after those bars. Eleven randomly selected vectors were used as the opposite class members (class 0) in the fuzzy k-nn classifier (where k=5 and the *fuzzifier*[1] m=2). The window widths were three bars for both windows. Training epochs run 100 times. Texture-change curve was created by calculating the bar based means of all 100 distributions.

The most persistent and intensive texture changes take place after the bar 407 and between the bars 105-212 where the main theme is strongly present. Unlike the other features, the rhythm set approach did not show any significant correlation ($p \gg 0.05$) with the texture classifier curve. However, there would be no need

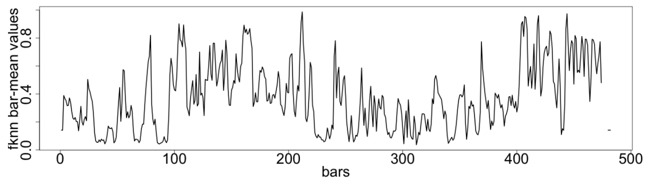

Figure 1. The intensity of the textural changes.

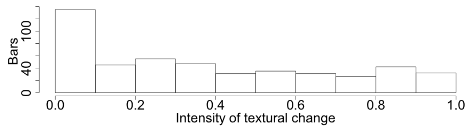

Figure 2. Histogram of textural changes.

for the traditional feature selection: On the contrary, if some of the features do not correlate with the resulting curve that gives information about the style of the composition and the selected feature vectors. The histogram confirms that textural change occurs at various intensities (almost evenly distributed) and that the texture is nearly constant for about 140 bars (see Figure 2).

There are still several open issues related to segmentation cardinalities and the selection of the class members which we expect to solve as we gain experience with the system.

4. References

Huovinen Erkki and Atte Tenkanen. 2008. Bird's Eye Views of the Musical Surface - Methods for Systematic Pitch-Class Set Analysis. *Music Analysis* 26(1-2).

Keller, J. M., Gray, M. R., and Givens, J. A. 1985. *A Fuzzy k-Nearest Neighbor Algorithm.* IEEE Trans. Syst. Man. Cybern., SMC-15(4): 580-585.

Lewin, David. 1980. A Response to a Response: On Pcset Relatedness. *Perspectives of New Music* 18(2): 498-502.

[1]The fuzzifier m (> 1) determines how the membership value varies with the distance which is weighted according to the rule $d(x, y)^{-2/(m-1)}$ (see Keller et al. 1985)