VOICE ENTRY ESTIMATION HEURISTICS TO REDUCE ERROR PROPAGATION IN VOICE SEPARATION MODELS

Reinier de Valk
Data Archiving and Networked Services (DANS)
reinier.de.valk@dans.knaw.nl

Tillman Weyde
City, University of London
t.e.weyde@city.ac.uk

ABSTRACT

In this paper we describe two voice entry estimation heuristics to reduce error propagation in voice separation models. Both heuristics estimate the entry points of the individual voices, by means of which a stable framework of reference chords can be constructed. The heuristics are integrated into a previously proposed neural network-based voice separation model. The extended model is evaluated on 78 keyboard works by Johann Sebastian Bach, and is shown to perform statistically significantly better than the original model on the complete dataset.

1. INTRODUCTION AND RELATED WORK

Within the domain of symbolic music representation, *voice separation* is defined as the task of separating a musical piece consisting of single- and multi-note sonorities into independent constituent voices [1]. A prerequisite for a number of MIR and musicological tasks (transcription, pattern retrieval), adequate voice separation has been recognised as an important but challenging task, for which no satisfactory solution has been proposed so far.

Existing models addressing the task can be divided into a variety of rule-based models, e.g., [2, 6, 10], some allowing non-monophonic voices, e.g., [9], and machine learning models, such as a decision tree [7], hidden Markov models, e.g., [3, 8], and neural networks, e.g., [3–5].

2. MODEL, FRAMEWORK, AND DATA

We formulate the task of voice separation as a multi-class classification problem, where each note must be assigned to one of v voices, the classes. As the classifier we use a standard three-layer feed-forward neural network with resilient backpropagation. The model, N, processes the music from left to right, starting at the lowest note of each chord. It takes as input a vector of hand-crafted features encoding properties of a single note in its polyphonic context. The model has been integrated in our framework for preprocessing, feature extraction, and cross-validated evaluation, implemented in Java.

The model is evaluated on Johann Sebastian Bach's 30 inventions and the 48 fugues from the *Well-Tempered Clavier*. The data, in MIDI format, was originally retrieved from the CCARH's MuseData repository, [1] and has been slightly adapted. Full details on problem formulation, model, framework, and data can be found in [3]. [2]

3. EVALUATION

We use *accuracy*, the percentage of notes assigned correctly, as the main metric to assess model performance. Furthermore, we use two evaluation modes: *test mode*, where the feature vectors are calculated using the correct voice information for the preceding notes, and *application mode*, where the feature vectors are calculated using the model-generated voice information.

In application mode, the model is susceptible to *error propagation*, i.e. incorrect voice assignments entailing more incorrect assignments. Given the accuracy in test (acc_T) and application mode (acc_A), the proportion of misassignments due to error propagation, q, is as follows:

$$q = \frac{\text{acc}_\text{T} - \text{acc}_\text{A}}{1 - \text{acc}_\text{A}}. \quad (1)$$

4. VOICE ENTRY ESTIMATION HEURISTICS

In previous work [3, 4] we observed that model performance is strongly affected by error propagation. Although error propagation can occur throughout a piece, it tends to be particularly strong in thinly-textured openings of pieces, where the model tends to start 'on the wrong foot'—often leading to numerous misassignments.

To address this problem, we propose two heuristics, h1 and h2. They estimate which notes belong to the *new voices* at each *density increase*, i.e., point where the maximal number of simultaneous notes so far increases. h1 and, partly, h2 are based on the prior assumptions that (i) voices tend to move in small steps, and that when new voice(s) enter, (ii) none of the already active voices has a rest, and (iii) none of the voices is involved in voice crossing.

h1 and h2 share a similar overall algorithmic structure, as shown in Figure 1. The algorithm takes as input a list

[1] See `http://www.musedata.org/`.

[2] The code and data can be downloaded from `http://mirg.city.ac.uk/datasets/rdv/phd_thesis/`.

```
01  function estimate(list notes) returns list
02    density increases d := [d_1, ..., d_m]
03    available voices av := [1, ..., d_m]
04    add av to new list fw
05    for i from m to 2:
06      if h1: find lowest-cost configuration
07        at pos(d_i)
08      if h2: find pattern at pos(d_i)
09      remove new voices from av
10      prepend av to fw
11      av := copy(av)
12    return fw
```

Figure 1. Algorithm outline. Underlined concepts are explained in the main text.

of notes representing a piece, and returns, for each density increase (including the opening), a vector of voice assignments for the first chord of increased density. If the voices enter successively, h2 is called; if not, or if h2 fails, h1.

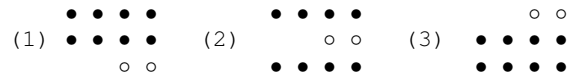

Figure 2. Chord configurations ($n = 2$). Columns represent chords; rows represent layers.

h1 is the more generic heuristic. It determines the new voices by calculating, at each density increase starting at the last, the lowest-cost *configuration*. A configuration organises the last n chords (i.e., ordered sequences of pitches) of density d_{i-1} and the first n chords of density d_i into horizontal layers, as shown in Figure 2. The cost for a configuration is calculated as follows:

$$\sum_{j=1}^{n} \sum_{k=1}^{d_{i-1}} \sum_{l=1}^{n} |p_{j,k} - p_{l,k}|, \qquad (2)$$

where $p_{j,k}$ is the pitch at the kth $2n$-sized layer in the jth chord of density d_{i-1}, and $p_{l,k}$ the pitch at the kth $2n$-sized layer in the lth chord of density d_i. The positions of the remaining n-sized layers in the lowest-cost configuration, then, determine the new voices.

h2 caters specifically to imitative pieces, and attempts to determine the new voices by finding, at each density increase starting at the last, a match for the *pattern* (as defined by the first n notes of the piece) in the first n chords of density d_i. The first matching criterion is rhythmic sequence; if this yields multiple matches, melodic contour is added as a second matching criterion. If a single match is thus found, the new voice is identified; if multiple matches are still found, the lowest-cost configuration (as in h1) is used to disambiguate. If no match is found, h2 fails.

5. DISCUSSION

Table 1 shows that the extended model (N/h1 or N/h2) always outperforms the original model; the decrease in q corroborates that the heuristics successfully reduce error propagation. A test for statistical significance (Wilcoxon signed-rank test; significance criterion $p < 0.05$) shows that on the three- and four-voice fugues, N/h1 and N/h2 perform significantly better than N, and that on the four-voice fugues, N/h2 performs significantly better than N/h1. On the complete dataset, the performance improvement (N-N/h1; N-N/h2; N/h1-N/h2) is always significant.

Dataset		N		N/h1		N/h2	
	acc_T	acc_A	q	acc_A	q	acc_A	q
I (2vv)	99.63	98.76	0.70	**98.80**	0.69	**98.80**	0.69
I (3vv)	98.42	94.17	0.72	**94.22**	0.72	**94.22**	0.72
F (2vv)	99.38	96.54	0.82	**99.50**	n/a	**99.50**	n/a
F (3vv)	98.62	92.49	0.81	94.50	0.74	**94.58**	0.74
F (4vv)	97.57	80.70	0.87	85.76	0.82	**87.20**	0.81
F (5vv)	90.12	63.63	0.72	**67.63**	0.69	65.07	0.71
all	98.18	89.28	0.83	91.73	0.78	**92.12**	0.76

Table 1. Results ($n = 3$). I = inventions; F = fugues.

A strict comparison with the state-of-the-art is difficult due to the heterogeneity of datasets and metrics used; nevertheless, the extended model also outperforms results reported (see Section 1).

6. REFERENCES

[1] E. Cambouropoulos. Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception*, 26(1):75–94, 2008.

[2] E. Chew and X. Wu. Separating voices in polyphonic music: A contig mapping approach. In U. K. Wiil, editor, *Computer Music Modeling and Retrieval*, pages 1–20. Springer, 2005.

[3] R. de Valk. *Structuring lute tablature and MIDI data: Machine learning models for voice separation in symbolic music representations.* PhD thesis, City, University of London, 2015.

[4] R. de Valk and T. Weyde. Bringing 'Musicque into the tableture': Machine-learning models for polyphonic transcription of 16th-century lute tablature. *Early Music*, 43(4):563–576, 2015.

[5] P. Gray and R. Bunescu. A neural greedy model for voice separation in symbolic music. In *Proc. 17th ISMIR*, pages 782–788, 2016.

[6] N. Guiomard-Kagan et al. Improving voice separation by better connecting contigs. In *Proc. 17th ISMIR*, pages 164–170, 2016.

[7] P. B. Kirlin and P. E. Utgoff. VoiSe: Learning to segregate voices in explicit and implicit polyphony. In *Proc. 6th ISMIR*, pages 552–557, 2005.

[8] A. McLeod and M. Steedman. HMM-based voice separation of MIDI performance. *Journal of New Music Research*, 45(1):17–26, 2016.

[9] D. Rafailidis et al. Musical voice integration/segregation: VISA revisited. In *Proc. 6th SMC*, pages 42–47, 2009.

[10] D. Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, 2009.