Chapter 6
Chord- and Note-Based Approaches to Voice Separation

Tillman Weyde and Reinier de Valk

Abstract Voice separation is the process of assigning notes to musical voices. A fundamental question when applying machine learning to this task is the architecture of the learning model. Most existing approaches make decisions in note-to-note steps (N2N) and use heuristics to resolve conflicts arising in the process. We present here a new approach of processing in chord-to-chord steps (C2C), where a solution for a complete chord is calculated. The C2C approach has the advantage of being cognitively more plausible but it leads to feature modelling problems, while the N2N approach is computationally more efficient.

We evaluate a new C2C model in comparison to an N2N model using all 19 four-voice fugues from J. S. Bach's *Well-Tempered Clavier*. The overall accuracy for the C2C model turned out slightly higher but without statistical significance in our experiment. From a musical as well as a perceptual and cognitive perspective, this result indicates that feature design that makes use of the additional information available in the C2C approach is a worthwhile topic for further research.

6.1 Introduction

Voice separation is the process of assigning notes to a musical voice. This is not only an important feature of our auditory and specifically musical perception and cognition, but also a practical problem when working with semi-structured musical data, such as tablature, MIDI recordings, or automatically transcribed audio. When using any of these forms of input data, automatic voice separation (AVS) is an important part of creating musical scores together with pitch estimation and beat tracking. AVS has not achieved as much attention as some other tasks in music information retrieval, such as genre classification or melodic similarity estimation. This is possibly because

Tillman Weyde · Reinier de Valk
Department of Computer Science, City University London, London, UK
e-mail: {t.e.weyde, r.f.de.valk}@city.ac.uk

D. Meredith (ed.), *Computational Music Analysis*,
DOI 10.1007/978-3-319-25931-4_6

polyphonic transcription from audio has not reached the level of accuracy needed for practical software applications. However, in applications with MIDI and tablature AVS can be useful immediately.

Several algorithmic methods have been developed, using different approaches. Almost all existing voice separation models are based on the Gestalt principle of proximity in pitch and time. This leads to a well-defined solution for simple cases, but in practice the complexity of real music requires trade-offs between many different factors such as time and pitch proximity. This creates two challenges. First, it is not obvious how to weight different features to achieve decisions in situations with conflicting cues. This problem can be addressed with machine learning by adapting the weights to fit the data, e.g., using a neural network as in the work presented in this chapter. Second, there are many possible assignments of notes to voices, so that the implementation of an exhaustive search of all possible combinations of note-to-voice assignments across all notes in a piece is computationally intractable. Therefore, a localized modelling of the decision process is necessary. We describe here a new chord-to-chord (C2C) approach to voice assignment, with two different feature extraction methods, and compare it to the previously introduced note-to-note (N2N) approach. From a computational perspective, the C2C approach has the advantage over existing note-based models that notes can be assigned to voices even where the individual assignment is not optimal (e.g., in terms of pitch distance), if it leads to a better fit for the whole chord. The different approaches exhibit differences in terms of their runtime complexity and performance and entail different trade-offs.

From a perceptual and cognitive perspective, the separation and integration processes in forming multiple voices or streams of auditory events are an example of the binding problem, i.e., the integration of stimuli, like notes, to percepts, like voices (cf. Wrigley and Brown, 2004). These binding mechanisms in general and for music in particular are not yet well understood, but it is clear that parallel processing is involved. The C2C approach includes that aspect in the model, increasing its power and cognitive plausibility. However, the C2C approach poses the problem of how to model the relation between the notes and the voices and their combinations, both from a perceptual and cognitive as well as from a machine learning perspective.

In the N2N approach, the concepts of pitch and time proximity can be modelled directly, because only one note is treated per decision, so that the time and pitch distance to notes in previous voices can be directly used as feature values. In the C2C approach, however, the modelling of variable chord sizes with a vector of fixed dimension is an open problem. Conversely, in the C2C approach the interaction between notes in the same chord is naturally modelled in the voice assignment and learning process, while the N2N approach requires the use of additional rules and heuristics and cannot guarantee optimal results on the chord level. We have performed experiments on the four-voice fugues in J. S. Bach's Well-Tempered Clavier, testing implementations of the N2N and C2C approaches in different variants.

In Sect. 6.2, we give an introduction to AVS and related work in this area. In Sect. 6.3, the application of machine learning to voice separation, feature design, and the implications of the C2C and N2N model are discussed. The experiments we

conducted are described in Sect. 6.4 and the results are discussed in Sect. 6.5. The final conclusions and directions for further work are presented in Sect. 6.6.

6.2 Voice Separation

Voice separation has been approached both from the perspective of music perception and cognition, in particular with respect to the underlying neural mechanisms, as well as from a computational perspective, with a focus on practical problem solving.

6.2.1 Perception and Cognition of Polyphony

Several methods for voice separation in symbolic music formats have been suggested in recent decades. Earlier work in the 1980s and 1990s focused on modelling cognitive and perceptual phenomena relating to polyphonic structure. Specifically the work of Huron (1989, 1991b) addressed the perception of polyphony, prevalence of voice crossings, and inner-voice entries in corpora (Huron, 1991a; Huron and Fantini, 1989), presenting a model for measuring pseudo-polyphony; Marsden (1992) developed rule-based models for the perception of voices in polyphonic music. Gjerdingen (1994) used a neural network to model the perception of 'apparent motion'.

Auditory streaming, and more generally *auditory scene analysis* as defined by (Bregman, 1994), i.e., the integration and separation of auditory stimuli into separate streams, representing environmental properties (sounds originating from the same source), is generally assumed to be the perceptual and cognitive basis for the development of musical polyphony (e.g., Huron, 1991a; Wright and Bregman, 1987). McCabe and Denham (1997) presented a model of the early stages of the process of auditory streaming on the acoustic level, which is not directly applicable to symbolic voice separation. Perception of separate voices has been shown to take place in musically trained and untrained subjects, and to depend on sensory-driven and attention-driven processes (Fujioka et al., 2008, 2005). Both the temporal processes and concurrent stimulus processing in chord perception interact, and this interaction is not fully understood (McDermott and Oxenham, 2008; Turgeon and Bregman, 2001). The neural mechanisms underlying these functions are the topic of current research in auditory neuroscience (see, e.g., Ragert et al., 2014; Shamma et al., 2011).

Modelling the perception of multiple voices in polyphonic music poses an instance of the binding problem, i.e., the segregation and combination of different stimuli into a perceptual entity, and it is not fully understood how this is represented or processed in the brain. In particular the combinatorial representation of auditory features and objects (voices) poses modelling questions: assuming specific neurons representing possible combinations would require many representation units, which would have very sparse activations, and thus be ecologically implausible. Many

researchers assume that temporal synchronization mechanisms play a role (Brown et al., 1996; Shamma et al., 2011).

6.2.2 Automatic Voice Separation

Apart from being of importance for understanding underlying musical principles and perceptual and cognitive aspects of polyphony, AVS is necessary for music transcription and music information retrieval tasks on recorded MIDI data or automatically transcribed audio, where no score is available. Therefore several practical approaches have been proposed.

Over the past 15 years, several computational methods for voice separation have been developed, most of them rule-based like the earlier work by Marsden (1992). Temperley (2001) adopted an approach based on four 'preference rules', i.e., criteria for evaluating a possible analysis. Two of these match the Pitch Proximity Principle and the Principle of Temporal Continuity (Huron, 2001), which dictate that the closer two notes are to one another in terms of pitch and time, the more likely they are perceived as belonging to the same voice; the other two aim to minimize the number of voices (New Stream Rule) and to avoid shared notes (Collision Rule). Cambouropoulos (2000) briefly described an elementary version of a voice separation algorithm based on path length minimization. Chew and Wu (2004), and an extended method by Ishigaki et al. (2011), used a *contig* approach, in which the music is divided into segments where a constant number of voices is active (the contigs). The voice fragments in the segments are then connected on the basis of pitch proximity, disallowing voice crossings. Szeto and Wong (2006) considered voices to be clusters containing events proximal in the pitch and time dimensions, and model voice separation as a clustering problem. The aim of their research, however, was to design a system for pattern matching, not voice separation. In their method, voice separation is only a preprocessing step that prevents "perceptually insignificant" (p. 111) stream-crossing patterns from being returned by the system. Kilian and Hoos (2002) presented an algorithm that is not intended primarily for correct voice separation, but rather for creating "reasonable and flexible score-notation" (p. 2), so that their method allows for complete chords in a single voice. Similarly, in the method presented by Karydis et al. (2007), a voice is also not necessarily a monophonic sequence of notes. Rather, they preferred to use the term 'stream' (see Cambouropoulos, 2008) as a perceptually independent sequence of notes or multi-note sonorities. Hence, in addition to the 'horizontal' pitch and time proximity principles, they included two 'vertical integration' principles, based on principles suggested by Huron (2001), into their method: the Synchronous Note Principle (based on Huron's Onset Synchrony Principle) and the Principle of Tonal Fusion (based on Huron's Tonal Fusion Principle). A related algorithm is described by Rafailidis et al. (2009). Madsen and Widmer (2006) present an algorithm based fundamentally on the pitch proximity principle, which combines rules, optimization, and heuristics.

Both the rule-based and the machine learning algorithms that are described below treat the music as a set of events in a two-dimensional pitch-time grid. To search the full space of all possible voice assignments on this view would mean to explore all possible combinations of assignments of notes to voices. Thus the total number of combinations is V^n, where V is the number of voices and n is the number of notes. For example, for a relatively short piece with $V = 4$ and $n = 150$ this yields $4^{150} = 2^{300} \approx 2 \times 10^{90}$, making a complete search intractable. One principled approach to addressing this issue is the use of dynamic programming, which reduces the search space to a series of individual decisions, such that the overall result is optimal. However, in order to apply dynamic programming, the problem formulation must conform to Bellman's *principle of optimality* (Bellman, 2003) and decompose into *overlapping subproblems* (Cormen et al., 2009). Dynamic programming is used, for example, in Temperley's voice separation system (Temperley, 2001). The Viterbi algorithm is also based on dynamic programming and is used with hidden Markov models, which we have applied to AVS in earlier work (De Valk et al., 2013). However, the use of dynamic programming limits the modelling flexibility. Therefore the rule-based methods listed above reduce the complexity by applying specifically designed techniques that are based on decisions made per note or per note-pair within a limited context. They use heuristics and conflict resolution techniques to make sure that constraints are respected and some interaction between the note-level decisions is realized.

Machine learning approaches, in contrast, normally use a reduced representation, typically vectors of real numbers, that fits the requirements of the chosen learning technique. Kirlin and Utgoff (2005) consider only pairs of notes within a window, and use decision tree learning to determine whether or not they belong to the same voice. Assigning notes to voices in order of their onsets, they use pitch proximity to disambiguate between multiple voices. Within a chord (notes with common onset time), only one note is considered at a time and all possible next notes within the voice are evaluated. The complexity is further reduced by limiting the window size and using the pitch proximity heuristic.

Jordanous (2008) also adopts a machine-learning approach. In her method, within windows based on instances of the maximal number of voices present, every note is compared to every possible next or previous note (the method starts from a marker, which defines the middle of the window). Using learned pitch transition probabilities and voice probabilities given a pitch, the most probable voice for each note is determined. As in Kirlin and Utgoff's algorithm, for each note within a chord, all pairs that it forms with possible next and previous notes are considered, with conflict resolution based on the highest probabilities.

In neither Kirlin and Utgoff's (2005) nor Jordanous' (2008) machine learning approaches is there any optimization of the assignment of notes to voices for a chord as a whole.

6.3 Method

We approach the problem of AVS in a standard machine learning set-up. AVS on symbolic music representations operates on a complex input consisting of a stream of often synchronous note events, and machine learning offers the possibility for a system to adapt its behaviour to data. By using given voice information, e.g., as given by the composer or a human transcriber, the assignment of the events to voices can be seen as a supervised machine learning problem. The given voice information, the so-called *ground-truth*, defines the target output of the system.

Most machine learning models use a vector of numbers as input and output a vector of numbers or symbols. An important question is how the properties of the musical decision context can be encoded in a vector, so that the relevant aspects of time and pitch structure are reflected and usable for the learning model (dynamics, timbre and expressive timing are not included in the model at this point). In the case of voice separation, a straightforward approach is the N2N model, where the context is modelled for each note using a vector of features such as the pitch distances to the previous notes in the voices (a detailed description follows in the next section). Based on this feature vector, a decision to assign the current note to a voice is made and the following steps are based on this decision—that is, we follow a greedy approach. The advantage is that the system is conceptually straightforward and has a computational complexity that is linear in the number of notes processed. The disadvantage is that the solution may be sub-optimal, because decisions are not revised. For example, in some cases, it may be better *not* to minimize the proximity of the current note within its voice, but rather go for another option if that enables a better fit for the remaining notes in the chord. In the N2N approach such a solution cannot be found. The N2N approach has nevertheless been applied using neural networks with good results, but also with potential for improvement by supporting chord-level optimization.

We therefore propose our new C2C approach for modelling voice separation as a machine learning problem on the level of chords rather than notes, where a non-linear rating function that maps notes to voices per chord is learned. The C2C approach is still local in time but global in the vertical dimension—that is, we evaluate all possible assignments of notes to voices. This approach is cognitively and perceptually more plausible, as it addresses the concurrent processing that takes place in the human auditory system. The binding problem, however, also applies in this context, as the machine learning process requires a compact representation, which also helps to keep the model plausible. This approach requires the design of features representing complete chords rather than just notes, for which we use two approaches.

The first is to represent each voice separately with a set of features, raising the problem of how to represent voices without notes with a fixed-size vector. The second is to average the feature values over the notes in the chord, avoiding the problem of missing notes and leading to a compact representation with fixed dimensions, but also to a loss of information. Within this framework we are not aware of a solution that would avoid these problems.

6.3.1 Representation

As discussed above, most methods for voice separation focus on features of the music that represent the proximity of notes in pitch and time. This can be measured, for example, as the pitch difference from the note to be assigned to the previous note in each voice. Similarly we can measure the time from the end of the previous note in each voice to the beginning of the current note. We can then choose the voice that minimizes one of these distances or a combined distance, but that may not be the optimal solution for a whole chord, which requires the definition of a global measure. Before we discuss the question of how to determine the best trade-off for a given chord and mapping, we present the feature definitions that we use in the C2C and the N2N approaches.

6.3.1.1 Features

We define an n-dimensional feature vector as a numerical representation of the notes in a chord in their polyphonic context. n is fixed within a learning and application model, but depends on the maximum number of voices V the model supports. Pitches are represented as MIDI note numbers, pitch intervals as semitones, and durations as whole notes. When using a model designed for V voices, a chord can contain 1 to V notes.

The feature vector has three parts:

1. *note-specific features* are different for the individual notes in the chord. Each of these gets a default value of -1 for each note the chord is short of V, that is, for notes c to $V-1$, where c is the number of notes in the chord (using zero-based indexing);
2. *chord-level features* are calculated per chord;
3. *polyphonic embedding features* depend on the mapping of notes to voices for the chord; each mapping results in a different polyphonic embedding.

Let n_t^v be the chord note mapped to voice v under the current mapping, n_{t-1}^v the previous note in v, $\mathtt{p}(n)$ a note's pitch, $\mathtt{on}(n)$ its onset time, and $\mathtt{off}(n)$ its offset time. For each voice v we calculate

- the pitch proximity of n_t^v to n_{t-1}^v:

$$\mathtt{pitchProx}(v) = \frac{1}{|\mathtt{p}(n_t^v) - \mathtt{p}(n_{t-1}^v)| + 1}\,; \tag{6.1}$$

- the inter-onset time proximity of n_t^v to n_{t-1}^v:

$$\mathtt{intOnProx}(v) = \frac{1}{(\mathtt{on}(n_t^v) - \mathtt{on}(n_{t-1}^v)) + 1}\,; \tag{6.2}$$

- the offset-onset time proximity n_t^v to n_{t-1}^v:

$$\texttt{offOnProx}(v) = \begin{cases} \frac{1}{(\texttt{on}(n_t^v) - \texttt{off}(n_{t-1}^v)) + 1}\,, & \text{if } \texttt{off}(n_{t-1}^v) \leq \texttt{on}(n_t^v)\,, \\ \frac{1}{(\texttt{on}(n_t^v) - \texttt{off}(n_{t-1}^v)) - 1}\,, & \text{if } \texttt{off}(n_{t-1}^v) > \texttt{on}(n_t^v)\,; \end{cases} \tag{6.3}$$

- the pitch movements—that is, for each voice v, the difference $p(n_t^v) - p(n_{t-1}^v)$, in semitones;
- the pitch–voice correlation ρ between the chord's pitch ordering and voice ordering, as measured by the Pearson correlation coefficient:

$$\rho_{pv}(C) = \frac{\sum p_i v_i - c\bar{p}\bar{v}}{s_p s_v} = \frac{\sum p_i v_i - \sum p_i \sum v_i}{\sqrt{\sum p_i^2 - (\sum p_i)^2}\,\sqrt{\sum v_i^2 - (\sum v_i)^2}}\,, \tag{6.4}$$

  where c is the number of notes in the chord C, p_i the pitch of note i, and v_i the voice assigned to note i. If there is only one note, 0 is returned.

The decision to model proximity as $1/\text{distance}$ rather than using distance was taken in order to emphasize differences between smaller distances.

The whole feature vector is described in Table 6.1 for the C2C model and in Table 6.2 for the N2N model. Features marked with an asterisk (*) are assigned a value of -1 for every note the chord is short of V.

6.3.1.2 Variable Chord Sizes

A central problem in the C2C approach is that chords are of variable size, and standard neural networks, like most machine learning algorithms, use fixed-size vectors. We consider two approaches here for pitch and time proximity:

1) representing each voice separately, using default values when there are no notes in a voice;
2) averaging values over notes.

Approach 1 has the advantage of capturing all information in the voice assignment. However, when not all voices are present, the voice features are filled with default values. These values are outside the regular range of values, but this information is not explicit to the neural network (or any other vector-based learning system used). Therefore the learning system needs to learn the relation between the appearance of the default values and the rating of the mapping in context.

6.3.2 Learning Model

We have previously addressed the task of AVS with the N2N approach. The N2N approach models the task as a classification problem, where each note is assigned to a voice (a class). For each note in the dataset, a training example is created consisting of a feature vector and a ground-truth label (a one-of-n representation of the note's

Table 6.1 The feature vector for the C2C model, assuming $V = 4$

Position	Name	Description
Note-specific features		
1, 5, 9, 13	`indexInMapping*`	index (based on pitch) in the chord, excluding sustained previous notes
2, 6, 10, 14	`indexInChord*`	index (based on pitch) in the chord; including any sustained previous notes
3, 7, 11, 15	`pitch*`	pitch
4, 8, 12, 16	`duration*`	duration
Chord-level features		
17	`chordSize`	number of notes in the chord; including any sustained previous notes
18	`isOrnamentation`	true (1) if a sixteenth note or smaller and the only new note in the chord
19	`metricPosition`	metric position within the bar
20–22	`intervals*`	intervals in the chord; including any sustained previous notes
Polyphonic embedding features		
23–26	`pitchProx`	for each voice v: the pitch proximity of n_t^v to n_{t-1}^v
27–30	`intOnProx`	for each voice v: the inter-onset time proximity of n_t^v to n_{t-1}^v
31–34	`offOnProx`	for each voice v: the offset-onset time proximity of n_t^v to n_{t-1}^v
35–38	`pitchMovements`	for each voice v: the pitch movement of n_t^v with respect to n_{t-1}^v
39–42	`voicesAlreadyOccupied`	binary vector encoding all voices already occupied in the chord
43	`pitchVoiceRelation`	correlation coefficient between pitches and voices; including any sustained previous notes
44	`numberOfVoiceCrossingPairs`	number of voice crossing pairs; including any sustained previous notes
45	`summedDistOfVoiceCrossingPairs`	total distance of all voice crossing pairs; including any sustained previous notes
46	`avgDistOfVoiceCrossingPairs`	average distance of all voice crossing pairs; including any sustained previous notes
47–50	`mapping`	vectorial representation of the mapping of notes to voices

voice). Given the feature vectors as input, the network is trained so that its output approximates the labels. The trained network is then applied to unseen data to predict voices; the predictions are determined by the output neuron with the highest activation. More details on this approach are provided by De Valk and Weyde (2015).

In the C2C approach, the task is modelled as a regression problem, where the ratings for the mappings of notes to voices are learned with the aim of giving the ground-truth mapping the highest rating. Each chord in the dataset is represented as a set of m n-dimensional feature vectors, where n depends on the selected feature set, voice number and use of averaging. Each vector encodes properties of that chord in its polyphonic context under one of m possible mappings of the chord notes to voices. The goal is to have a model that, for each chord in a piece, takes as input a set of feature vectors representing that chord in its context for all possible mappings of notes to voices, and that rates the correct mapping highest. We use a three-layer

Table 6.2 The feature vector for the N2N model, assuming $V = 5$

Position	Name	Description
Note-specific features		
0	`pitch`	pitch
1	`duration`	duration
2	`isOrnamentation`	true (1) if a sixteenth note or smaller and the only new note in the chord
3	`metricPosition`	metric position within the bar
Chord-level features		
4	`chordSize`	number of notes in the chord; including any sustained previous notes
5	`indexInChord`	index (based on pitch) in the chord; including any sustained previous notes
6	`pitchDistTo NoteBelow`	pitch distance to the note below in the chord; including any sustained previous notes
7	`pitchDistTo NoteAbove`	pitch distance to the note above in the chord; including any sustained previous notes
8–12	`intervals`	intervals in the chord; including any sustained previous notes
Polyphonic embedding features		
13–17	`pitchProx`	proximities in pitch to the previous note in each voice
18–22	`intOnProx`	proximities in time (inter-onset) to the previous note in each voice
23–27	`offOnProx`	proximities in time (offset-onset) to the previous note in each voice
28–32	`voicesAlready Occupied`	binary vector encoding all voices already occupied in the chord

feed-forward neural network model with an output layer containing only a single neuron. The activation value of this neuron ranges between 0 and 1, and represents the rating for a mapping given a chord and context. We use a sigmoid activation function and resilient backpropagation (RPROP) as the learning algorithm (Igel and Hüsken, 2003; Riedmiller and Braun, 1993).

6.3.3 Mappings: Enumeration and Pruning

In the C2C approach, the input to the neural network depends on the mapping of the chord notes to the voices. For each chord, given the number of notes in the chord and the number of voices in the dataset, all possible mappings of notes to voices are enumerated. A mapping is encoded as a V-dimensional vector, where V is the maximum number of voices and the ith element is an integer from -1 to $(N-1)$, where N is the maximum number of concurrent notes, which is normally equal to V. The element indicating the (zero-based) index into the chord of the note assigned to the ith voice (with the voices ordered from top to bottom and sustained previous notes excluded). A value of -1 at the ith position indicates that no note in the chord

is mapped to the *i*th voice. Note that the vectorial representation used does not allow for more than one note to be mapped to one voice (in which case the voice would no longer be monophonic). Thus there are initially V^{N+1} possible mappings of notes to voices.

The enumeration of mappings per chord is restricted according to the following constraints:

1. all chord notes are mapped to a voice;
2. each chord note is mapped to only one voice;
3. no chord note is mapped to a voice that is already taken by a sustained note from a previous chord.

We then prune the enumerations, both to avoid the method becoming computationally too expensive and to improve the model's performance. All mappings that contain more than two voice crossing pairs are removed, where voice crossing pairs are instances of voice crossings within a chord. If, for example, the tenor voice (T) moves above both the soprano (S) and the alto (A) in a chord (and the soprano is above the alto), this chord contains two voice crossing pairs: TS and TA. When counting voice crossing pairs, sustained notes from previous chords (if applicable) are included, and unisons are not considered to be voice crossings. The limit of two was chosen as this is the maximum number of voice crossing pairs encountered in our dataset. This is an ad hoc choice which matches the difficulty that listeners experience in perceiving voice crossings. However, whether this can be generalized and whether a more flexible solution can be found are questions that are worth exploring more thoroughly. Pruning can be very effective in reducing the number of mapping possibilities, as shown in Table 6.3, which gives the number of possibilities for a chord of c notes in a context of V voices before and after pruning (all numbers provided are for the case where there are no sustained notes from a previous chord). The table shows that the effect increases as the number of notes in a chord grows, and that it is stronger when there are more voices. When the number of notes approaches the number of voices, there are fewer options left within the given limit of voice crossings, leading to lower numbers of mappings (e.g., in the case $V = 5$, $c = 5$). In our experiments we used values of 4 and 5 for V, and we can read from the table the relative increase of computation compared to the N2N model. In the N2N model, c feature vectors need to be calculated and evaluated with the neural network. For the C2C model, the number in column P describes the number of feature vector calculations and neural network evaluations per chord after pruning. We can see that the increase for chords with more notes (i.e., P/c) is bounded by a constant factor of 7 for $V = 4$ and of 17 for $V = 5$. The feature vectors are larger for the C2C model, which adds additional processing, but in practice the overall increase was lower than predicted and below 11 (see Table 6.4), which is substantial but not prohibitive on modern computers.

As mentioned above, we used a neural network to calculate the gain function for each mapping, i.e., a rating of the quality of the mapping in the context. For learning from data we train the neural network to rate the mapping output (as in the ground-truth) better than any other mapping. For this we use relative training as used

Table 6.3 Number of mapping possibilities for a chord of c notes in a context of V voices, with no restrictions (N), meeting enumeration constraints (C) and after pruning (P) of mappings containing more than two voice crossing pairs

	$V=3$			$V=4$			$V=5$		
	N	C	P	N	C	P	N	C	P
$c=1$	3	3	3	4	4	4	5	5	5
$c=2$	9	6	6	16	12	12	25	20	20
$c=3$	27	6	5	64	24	20	125	60	50
$c=4$				256	24	9	625	120	45
$c=5$							3125	120	20

by Weyde and Dalinghaus (2003), which is based on Braun et al. (1991). Unlike a standard regression approach, where desired outputs—i.e., the ratings—are given directly, we train the model by comparing the output for the ground-truth mapping with the outputs for all other mappings, the goal being that, for each chord, the ground-truth mapping should lead to higher network output than the other mappings.

6.3.4 Relative Training

The relative training technique can be thought of as instantiating the network twice, sharing the structure and the weight values, with the ground-truth mapping input to one network and a generated mapping to the other. For training, both networks feed into a single *comparator* neuron through connections with fixed weights, -1 and $+1$, as shown in Fig. 6.1. In this structure, the feature vector of the ground-truth data is applied to 'net 1' and the feature vector of the other examples to 'net 2'. If net 1 with the ground-truth input has a higher output than net 2 (i.e., the relative rating is correct), then the input to the comparator becomes negative, otherwise the input value is positive, or zero if the networks produce equal outputs.

We used a rectified linear unit for the comparator neuron—that is, we used $\texttt{max}(x,0)$ as the activation function and set the target output of the comparator to 0, so that for all correct relative ratings (where $\texttt{out}(\text{net } 1) > \texttt{out}(\text{net } 2)$) the comparator output error $(\mathit{actual} - \mathit{target})^2$ is 0. This network can be trained with standard backpropagation or any other gradient descent algorithm to avoid or reduce incorrect relative ratings. It is helpful for new data to aim for an output difference that is not just negative, but below some threshold $-\varepsilon$. In neural network terminology, this means setting a bias for the comparator neuron. This introduces some robustness against variations in the input data and improves the performance on new data. An alternative, that was used by Hörnel (2004), is to use a logistic activation function $\frac{1}{1+e^{-x}}$, which leads to a positive error signal of varying size for all inputs. We did preliminary experiments with both approaches, and found that both approaches lead

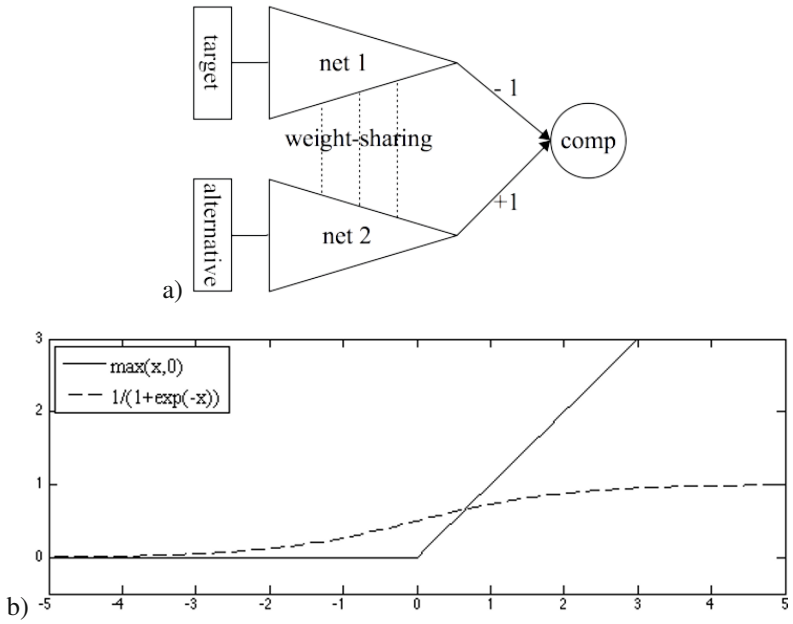

Fig. 6.1 (a) The neural network structure for relative learning, the comparator neuron is labelled 'comp'. (b) Activation functions for the comparator neuron: rectified linear ($\max(x,0)$) and logistic ($\frac{1}{1+e^{-x}}$)

to similar results. We chose the rectified linear units for the final experiments as they are computationally more efficient and lead to slightly better results on the test data.

6.4 Experiments

We evaluated the performance of the C2C model with the proximity features represented per-voice (C2C) and averaged (C2CA). We also evaluated the effect of additional—unused—voice features, by testing a five-voice version. For evaluation of the two approaches, the 19 four-voice fugues of J. S. Bach's *Well-Tempered Clavier* (BWV 846–893) were used. The MIDI encodings for the pieces from www.musedata.org were used and all notes beyond four simultaneous voices were removed (this is typically the final chord and a few notes just before the final chord). This decision was taken to enable comparison with the N2N model, which in its current implementation does not allow more than one note per voice at any given time. The models were trained and evaluated using 19-fold cross-validation, where each piece is one fold. We did not use random division of training samples, as samples from the same piece would contain regularities, possibly verbatim repetitions,

which would make the results unrepresentative for new music. By using piece-wise cross-validation we can use the results as estimates for results on unseen data of the same style.

Two modes of evaluation were used:

1. *test mode*, where the context features are calculated on the basis of the ground truth. This mode is useful for assessing the effectiveness of learning and generalization of the neural network to new data, without the effect of propagating errors; and
2. *application mode*, where the context features are calculated on the basis of the predicted voice assignments. This mode is like the intended use case, where no ground truth, i.e., no score, exists.

We used preliminary experiments to determine the values of the rating margin ε as 0.05 and weight decay λ as 0.00001 (C2C) and 0.00003 (N2N). The method showed little sensitivity to variations in these parameters, as long as ε was not much smaller and λ not much larger than the chosen values.

Five models were tested overall. The N2N model (see De Valk and Weyde, 2015), the C2C4 and C2C5 models without averaging of the proximity features, which have four and five voices, respectively, and the C2CA4 and C2CA5 with averaging and four and five voices, respectively.

6.5 Results and Discussion

Accuracy (number of notes correctly assigned, divided by total number of notes) was chosen as the evaluation metric. The accuracy of the C2C, N2N and C2CA models is shown in Table 6.4. The differences between the models' accuracies are not vast, but several differences are statistically significant. We used the Wilcoxon signed-rank test, a non-parametric test for different medians, to calculate significance. The test was applied to the accuracies per piece, thus with 19 samples per condition. As opposed to the frequently used Student's t-test, the signed-rank test does not assume the values to be normally distributed, and for the majority of the combinations in Table 6.4 the values' distributions are significantly non-normal according to a Jarque–Bera test.

Overall, the results are similar to previously reported results for the N2N model (De Valk et al., 2013), although somewhat lower, as this dataset contains no three-voice fugues. The test results are in a similar range to the training results, indicating no overfitting. The results in application mode are substantially lower than those in test mode, which is due to error propagation.

In the application mode, which represents the intended use case, the C2CA4 model is most accurate, improving over the N2N model by over 2 percentage points, but, as the standard deviation is high, the difference is not significant ($p = 0.11$). A general observation in the results is that the four- and five-voice versions of the C2C models perform similarly (as expected) and the differences between corresponding four- and five-voice models are not significant in the test and application modes.

Table 6.4 Voice assignment accuracy per model (columns) and mode (rows) with averages and standard deviations over 19 folds in cross-validation. The best average accuracies per mode are shown in bold. The running times on a four-core system are shown in the lower part of the table

		N2N	C2C4	C2C5	C2CA4	C2CA5
Training	Mean	**97.89%**	96.51%	96.20%	97.17%	97.03%
	Std-Dev	0.13	0.24	0.42	0.15	0.16
Test	Mean	**97.54%**	96.19%	96.17%	96.90%	96.79%
	Std-Dev	0.79	1.52	1.06	1.16	1.24
Application	Mean	77.70%	78.50%	76.91%	**80.24%**	78.31%
	Std-Dev	7.74	10.09	9.86	9.55	11.22
Running times (sec)	Training	1,498	11,579	12,608	11,005	11,777
	Test	48	485	503	493	507
	Application	55	306	313	308	320

The N2N model shows the highest accuracy in the training and test modes, which is significantly higher than the values for the C2C models. In test mode, the results are similar to training mode, but the standard deviation is greater and the difference between the N2N and C2CA models is not significant.

The averaging of the proximity features has a positive effect on the accuracy of the C2CA models compared to the standard C2C models. The difference between the different C2C and C2CA models is mostly significant for training and test mode, while in application mode only the difference between C2CA4 and C2C5 is significant.

Overall, the C2CA models perform slightly better than the N2N model in the application mode. In the training and test mode, the accuracy of the C2CA models is slightly lower than that of the N2N model, indicating that these models are not as flexible. This could be related to the loss of information due to averaging the proximity values, but the default values used in the C2C4 and C2C5 models seem to have an even worse effect and lead to lower accuracy in all cases.

Table 6.5 The significance p-values of differences between models according to a Wilcoxon signed-rank test. Above the diagonal is the test mode and below the application mode. Values shown in bold indicate significance at the 5% level. In the training mode, all models are significantly different from each other

	N2N	C2C4	C2C5	C2CA4	C2CA5
N2N		**.0001**	**.0000**	**.0008**	**.0004**
C2C4	.8596		.5412	**.0082**	**.0258**
C2C5	.4413	.1819		**.0003**	**.0039**
C2CA4	.1134	.4653	**.0062**		0.2753
C2CA5	.6507	.3736	.1336	.4180	

6.6 Conclusions

We introduced a new chord-to-chord (C2C) approach to automatic voice separation (AVS) in symbolic music representations. The C2C approach is based on processing the voice assignments for one chord as a whole, and calculating all possible assignments. This approach has the advantage of modelling interactions between notes in a chord and being perceptually and cognitively more plausible. Its disadvantage is that it is difficult to define the features effectively, where there is a trade-off between the problematic use of default values and the loss of information by averaging. In the presented evaluation, the differences in performance are small and vary between modes. The C2C model performs better than the N2N model when averaging the proximity features in application mode, which represents the use case. However, due to high variability, this result is not significant in our experiment. Given that the C2C model is clearly slower than the N2N model, the latter may be preferable for practical applications at the moment.

However, the results show that a C2C approach to voice separation can be successful. There is a need to understand better how to design effective chord-wise features for AVS. The temporal dynamics of the auditory scene analysis are only rudimentarily modelled here, and there may be better solutions to the representation and processing problem, for example using a time-based modelling approach (e.g., one based on synchronous activation of neurons). The development of better feature representations and effective global optimization techniques are thus relevant research topics for musical voice separation, which could help realize the potential of the C2C approach more fully in the future.

Acknowledgements Reinier de Valk is supported by a funded studentship from City University London and by a grant from Semantic Media Network/EPSRC.

References

Bellman, R. (2003). *Dynamic Programming*. Dover.

Braun, H., Feulner, J., and Ullrich, V. (1991). Learning strategies for solving the problem of planning using backpropagation. In *Proceedings of the Fourth International Conference on Neural Networks*, Nimes, France.

Bregman, A. S. (1994). *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press.

Brown, G. J., Cooke, M., and Mousset, E. (1996). Are neural oscillations the substrate of auditory grouping. In *ESCA Tutorial and Workshop on the Auditory Basis of Speech Perception, Keele University, July*, pages 15–19.

Cambouropoulos, E. (2000). From MIDI to traditional musical notation. In *Proceedings of the AAAI Workshop on Artificial Intelligence and Music: Towards Formal Models for Composition, Performance and Analysis*, Austin, TX.

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

Chew, E. and Wu, X. (2004). Separating voices in polyphonic music: A contig mapping approach. In Wiil, U. K., editor, *Computer Music Modeling and Retrieval*, volume 3310 of *Lecture Notes in Computer Science*, pages 1–20. Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

De Valk, R. and Weyde, T. (2015). Bringing 'musicque into the tableture': machine learning models for polyphonic transcription of 16th-century lute tablature. *Early Music*, in press.

De Valk, R., Weyde, T., and Benetos, E. (2013). A machine learning approach to voice separation in lute tablature. In *Proceedings of the Fourteenth International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 555–560, Curitiba, Brazil.

Fujioka, T., Trainor, L. J., and Ross, B. (2008). Simultaneous pitches are encoded separately in auditory cortex: an MMNm study. *Neuroreport*, 19(3):361–366.

Fujioka, T., Trainor, L. J., Ross, B., Kakigi, R., and Pantev, C. (2005). Automatic encoding of polyphonic melodies in musicians and nonmusicians. *Journal of Cognitive Neuroscience*, 17(10):1578–1592.

Gjerdingen, R. (1994). Apparent motion in music? *Music Perception*, 11(4):335–370.

Hörnel, D. (2004). Chordnet: Learning and producing voice leading with neural networks and dynamic programming. *Journal of New Music Research*, 33(4):387–397.

Huron, D. (1989). Voice denumerability in polyphonic music of homogeneous timbres. *Music Perception*, 6(4):361–382.

Huron, D. (1991a). The avoidance of part-crossing in polyphonic music: perceptual evidence and musical practice. *Music Perception*, 9(1):93–103.

Huron, D. (1991b). Tonal consonance versus tonal fusion in polyphonic sonorities. *Music Perception*, 9(2):135–154.

Huron, D. (2001). Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1):1–64.

Huron, D. and Fantini, D. (1989). The avoidance of inner-voice entries: Perceptual evidence and musical practice. *Music Perception*, 7(1):43–48.

Igel, C. and Hüsken, M. (2003). Empirical evaluation of the improved RPROP learning algorithms. *Neurocomputing*, 50:105–123.

Ishigaki, A., Matsubara, M., and Saito, H. (2011). Prioritized contig combining to segregate voices in polyphonic music. In *Proceedings of the Sound and Music Computing Conference*. Università di Padova.

Jordanous, A. (2008). Voice separation in polyphonic music: A data-driven approach. In *Proceedings of the International Computer Music Conference 2008*, Belfast, UK.

Karydis, I., Nanopoulos, A., Papadopoulos, A., Cambouropoulos, E., and Manolopoulos, Y. (2007). Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data. In *Proceedings of the 4th Sound and Music Computing Conference (SMC '07)*, Lefkada, Greece.

Kilian, J. and Hoos, H. H. (2002). Voice separation—A local optimization approach. In *Proceedings of the Third International Conference on Music Information Retrieval (ISMIR 2002)*, Paris, France.

Kirlin, P. B. and Utgoff, P. E. (2005). VoiSe: Learning to segregate voices in explicit and implicit polyphony. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, pages 552–557, London, UK.

Madsen, S. T. and Widmer, G. (2006). Separating voices in MIDI. In *Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR 2006)*, pages 57–60, Victoria, Canada.

Marsden, A. (1992). Modelling the perception of musical voices: a case study in rule-based systems. In Marsden, A. and Pople, A., editors, *Computer representations and models in music*, pages 239–263. Academic Press.

McCabe, S. L. and Denham, M. J. (1997). A model of auditory streaming. *The Journal of the Acoustical Society of America*, 101(3):1611–1621.

McDermott, J. H. and Oxenham, A. J. (2008). Music perception, pitch, and the auditory system. *Current opinion in neurobiology*, 18(4):452–463.

Rafailidis, D., Cambouropoulos, E., and Manolopoulos, Y. (2009). Musical voice integration/segregation: Visa revisited. In *Proceedings of the 6th Sound and Music Computing Conference*, Porto, Portugal.

Ragert, M., Fairhurst, M. T., and Keller, P. E. (2014). Segregation and integration of auditory streams when listening to multi-part music. *PloS one*, 9(1):e84085.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591.

Shamma, S. A., Elhilali, M., and Micheyl, C. (2011). Temporal coherence and attention in auditory scene analysis. *Trends in neurosciences*, 34(3):114–123.

Szeto, W. M. and Wong, M. H. (2006). Stream segregation algorithm for pattern matching in polyphonic music databases. *Multimedia Tools and Applications*, 30(1):109–127.

Temperley, D. (2001). *The Cognition of Basic Musical Structures*. MIT Press.

Turgeon, M. and Bregman, A. S. (2001). Ambiguous musical figures. *Annals of the New York Academy of Sciences*, 930(1):375–381.

Weyde, T. and Dalinghaus, K. (2003). Design and optimization of neuro-fuzzy-based recognition of musical rhythm patterns. *International Journal of Smart Engineering System Design*, 5(2):67–79.

Wright, J. K. and Bregman, A. S. (1987). Auditory stream segregation and the control of dissonance in polyphonic music. *Contemporary Music Review*, 2(1):63–92.

Wrigley, S. N. and Brown, G. J. (2004). A computational model of auditory selective attention. *IEEE Transactions on Neural Networks*, 15(5):1151–1163.