Learning Models for Interactive Melodic Improvisation

Belinda Thom
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.

Abstract. This research addresses the problem of the computer interacting with a live, improvising musician in the jazz/blues setting. We introduce BoB, a model of improvisation that enables the computer to trade solos with a musician in an adaptive, user-specific manner. We develop unsupervised learning methods for autonomously customizing the model via improvised examples and demonstrate the powerful musical abstractions that emerge when applied to Charlie Parker's *Mohawk* improvisations. Our key technical contribution is the development of an architecture that naturally enables unsupervised learned knowledge, perception and generation to be tightly coupled.

1 Introduction

This research addresses the problem of the computer interacting with a live, improvising musician in the jazz/blues setting. The long-term research goal is to create computer music algorithms and tools that enhance the organic, adaptive, melodic evolution that takes place when a soloing improviser *practices alone* with a fixed chord progression. Towards this end, we developed a model of improvisation that enables the computer to trade solos with a musician in an adaptive, user-specific manner.

This model is referred to as *Band-out-of-a-Box* (BoB) in tribute to the commercial software product *Band-in-a-Box* (BiB) [3]. Providing fixed-tempo accompaniment and/or melody/improvisation in the highly structured setting of a jam-session, BiB has been enormously successful in enhancing improvisational solo practice, despite the fact that during playback the musician is neither listened nor responded to. Furthermore, while a huge library of musical styles exists, user-specific customization of a musical style must be done up front, and relies entirely upon the user's musical expertise in the context of BiB.

Figure 1 introduces our BoB architecture. BoB operates in a similar setting to BiB; note the Lead Sheet, which structures the jam-session. However, BoB's focus is on human computer *interaction*, bringing the band – specifically, the improvising melodic companion –*out* of the box.

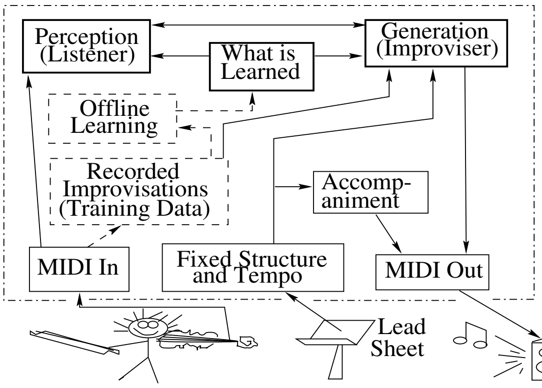


Figure 1: BoB Architecture

In this paper, we introduce BoB. We begin by describing the model's representation of perception and generation (bold lines), notable in that integration with the immediate musical environment is straightforward. We then focus on the *unsupervised* learning techniques (dashed lines) used to configure this representation with respect to a specific user, notable in that training data collection is effortless and noninvasive: merely observe what the user plays when they improvise (warmup). Finally, we present the learning results that are obtained by applying these techniques to Charlie Parker's *Mohawk* improvisations.

The most distinguishing and important aspect of this system is that *learning is used to tightly couple perception and generation.* In particular, with perception, learning provides the generalization needed to map a specific melodic fragment into a more abstract learned musical idea (many → one). With generation, the reverse takes place; a particular learned abstraction controls the stochastic search for an appropriate solo response (one → many).

Note: In our current implementation of BoB, real-time "MIDI In" is simulated using hand-entered transcriptions of Charlie Parker's improvisations [1]. Later, data will be obtained from a live musician via a MIDI controller.

2 The BoB Improvisational Model

Our improvisation model is inspired by the ideas of Johnson-Laird (1991) – improvised melodic response is spontaneous and highly constrained – a random search controlled by the musical sensibilities that committed practice produce. In BoB, committed practice takes the following forms:

- Domain knowledge, which constrains the representation and biases learning.
- Improvised example, both training data and/or the current evolving musical conversation.

2.1 Variable Length Trees

We introduce a new representation for solos, namely variable-length trees (VLT). VLTs are perceived and generated in real-time on a *bar-by-bar basis*. Variability refers to the fact that each bar (*context*) may contain a different number of notes.[1]

[1] In contrast to typical music learning, based on a fixed size input, VLT-based learning is more general.

Based on the content of Parker's transcriptions [1], VLTs were designed to encode and constrain melody as follows:

1. Rhythm is defined by the structure of the internal nodes; each must have 2 or 3 children.[2]
2. Pitch content, defined by octave and pitch class, is defined by the leaf nodes.
3. Melodic contour is defined by the leaves' ordering; identical consecutive pitches are tied unless specifically indicated.
4. A particular encoding must be as small (contain as few leaves) as possible.[3]

Figure 2 includes two contexts, A and B, and their equivalent VLTs. Superscript "-" distinguishes octave and subscript "." indicates no tie. Contexts A and B are two bars of solo taken from the *Mohawk* transcriptions.

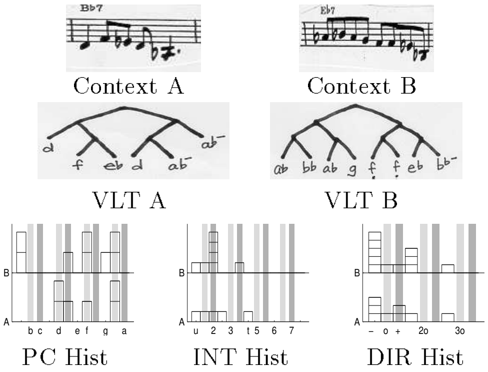


Figure 2: Example Representations

2.2 Local Histograms

By themselves, VLTs are simply glorified note sequences, albeit localized ones. While most prior work is ultimately based on a sequential viewpoint (e.g., [6], [7]), we seek a more abstract melodic representation in an attempt to capture deeper structure than is available on the immediate surface. The approach taken here stems from the belief that *the most salient aspect of melodic improvisation is not the trajectory of individual notes, but the average summary of local contexts of notes.*

Three types of VLT averages, or abstract feature sets are considered:

1. *Pitch Class* (PC): summarizes the frequency with which particular pitch classes occur.
2. *Intervallic Motion* (INT): summarizes the absolute interval between successive pitches.
3. *Melodic Direction* (DIR): summarizes the direction between successive pitch sequences.

In total, these summaries, recorded as histograms, capture a wide range of melodically pertinent information. That this information is *fully observable* means that it can be collected and processed by merely watching the user play. That these abstractions generalize – many sequences can map to the same histograms – is in part responsible for the following enormous real-time benefit: coarse estimation of duration and pitch (vs. exact transcription) may be sufficient for abstract data capture.

The abstract features for context A and B are shown in the bottom row of Figure 2. Each stacked box is a single count. The vertical columns (white, light gray, dark gray) provide visual aid for identifying consecutive columns. With PC, the x-axis contains 12 pitch classes, $[b\flat, b, c, d\flat, \ldots, a]$. With INT, x-axis values contain semitone differences, ranging from zero (u) to intervals greater than or equal to an octave. Numbers are shown for the Major intervals as well as the Perfect 5th; the tritone interval (t) is also displayed. With DIR, the x-axis is more complicated. The first 3 columns correspond to the number of times that *successive* pitches descend (-), remain identical (o) or ascend (+); the next three columns refer to the number of times that *3-note sequences* descend, remain identical (2o) or ascend; the final columns are analogous, but for *4-note sequences*.

One subtlety of the VLT representation is that notes which begin within up-beats (and/or end within down-beats) require successive identical pitched leaves (e.g., the $a\flat^-$ in context A). In other words, when summarizing pitch content, these notes receive additional counts – as perhaps these "syncopated" events should. Similar effects occur with INT and DIR.

2.3 VLT Generation

BoB needs to be able to create solos that respond in-kind to its user and environment. We developed an algorithm for generating VLTs as follows:

1. Select an initial tree (seed).
2. Randomly modify a population of trees according to some fitness.
3. Play the most fit tree that has emerged.

While the seed provides important links to the environment – e.g., use a VLT that approximates what the musician just played – and/or to domain knowledge – e.g., use a VLT from an appropriate part of the head – the most important aspect of this generation scheme is the intimate coupling that exists between it, machine learning algorithms, and our abstract feature sets.

For an abstract dataset, maximum likelihood (ML) techniques enable us to estimate (learn) what distribution over the features is most likely to have produced the data. In turn, this distributional information can be used to control the stochastic modification of VLTs. In particular, trees grow, collapse or modify a leaf or fringe of leaves at a time according to rolls of the die, which are in turn controlled by our parameter estimates. For example, if pitches were evolved in such a way that context A was very likely, we would expect the emergence of more notes of pitch class d and $a\flat$ in the population.

While the development of a stochastic modification procedure capable of transforming particular seeds into populations that adequately capture the salient aspects of *all three* interrelated

[2] which enables all note sequences but Parker's infrequent use of the pentuplet

[3] which guarantees one-to-one encodings

feature sets is a daunting task, the extremely local nature of our representation gives reason for optimism. For example, typical bar-based Parker VLTs contain approximately 10 leaves; that several appropriately distributed trees of comparable size can emerge through constrained, randomized search seems reasonable. It is also reasonable to expect that more direct rule-based approaches can also be used to aid in mapping from nontemporal histograms into musically pleasing, temporal note sequences.

Specifics regarding generation are outside the scope of this paper. It is worth noting that in a previous system implementation [8], acceptable levels of musical competence were displayed, given a minimally *hand-tuned* (vs. learned) configuration. 2 ms per-bar generation was achieved on a Pentium/NT platform using a population with 80-members, modified over 8 iterations.

2.4 Learning from Abstract Datasets

Rather than estimate a single probability distribution that best describes an abstract feature set, we assume that a *mixture* of k different distributions (components) generated the data – a distinction with enormous impact. For example, only the most mundane musician would play notes with the same likelihood throughout their entire performance. Surely, one thing that makes an improvisation *interesting* is that different pitches are preferred at different times – which relates to how various scales are utilized as the improvisation unfolds.

Yet, as in many statistical analyses of music [5], typical histogram-pitch-based representations rely on temporally global, single components, i.e., $k = 1$, with data averaged over an entire piece. While Dannenberg and Mont-Reynaud (1987) treat time locally – a different histogram is built for each eighth note of the 12-bar blues progression – a single component is still used. As the authors note a need for a less disperse likelihood function, it may be worthwhile to consider averaging slightly larger contexts, e.g., 1 bar, given an ability to account for several different internal modes of playing.

This mixture-based method of learning is *unsupervised* because we are not told which histograms were generated by what components. As such, state is hidden and only locally optimal likelihood estimates are guaranteed. With hidden state, learning involves estimating:

1. Parameters for k component distributions, which enables the generation of novel, like-minded histograms.
2. The odds that each training histogram was generated by a particular component. With this information, histograms can be clustered into 1 of k groups, in essence a more abstract form of musical perception.[4]

ML estimation in this domain is non-trivial. For one thing, because histograms may be comprised of different numbers of counts, those with larger sample sizes need to be more heavily weighted. In addition, our feature sets are nominal and should also be treated as unordered – who,for example, is to say when pitch class f is closer to e than to d? As detailed in [9] these issues are accommodated by basing components on variable-sized multinomial distributions. And, with this approach, the data determines what notions of nominal proximity to use.

2.5 Learning Parker's *Mohawk*

Experiments were conducted in order to investigate our ability to learn multinomial mixtures for these abstract histograms, using Parker's *Mohawk* improvisations as training data. Two transcriptions were used – *Mohawk I*, displayed in Figure 3 as choruses 1-5[5] and *Mohawk II*, displayed as choruses 6-10. Each of these improvisations is based on the B♭ 12-bar blues progression. While choosing the "appropriate" k for a given feature/data set is problematic, in our experiments, we used the largest value for which repeated searches produced consistent clusterings.

The symbols in Figure 3(a) identify which of the estimated PC distributions shown in Figure 3(c) are most likely ($k = 4$). In Figure 3(c), features with higher bars are more probable (their cumulative sum adds to one). In terms of each component's pitch preferences, this mixture can be viewed as an instantiation of the following scales:

Comp	Scale	n	Symbol (Box)
4	E♭ Bebop Dom	41	small black
3	B♭ Maj	11	white
2	E♭ Maj	28	gray
1	B♭ Bebop Maj	39	large black

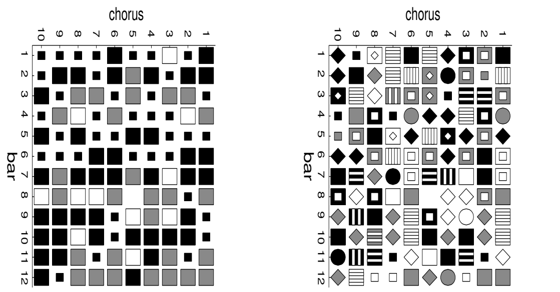
The n column indicates how many bars were assigned to each component. As is evident in Figure 3(c), Parker's realization of the B♭ Maj scale also contains a fair bit of chromaticism. The emergence of musically insightful abstractions adds credibility to this approach:

- The I and IV Bebop scales are most common.
- In bars 5,6 Bebop scales are used exclusively.
- In bars 9,10, the B♭ Bebop scale dominates.
- The E♭ Maj scale dominates the turnbacks.

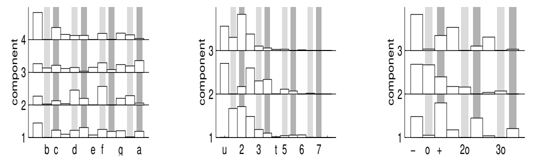
Multinomial components were also independently learned for the INT and DIR feature sets (Figures 3(d) and (e); for each, $k = 3$). Musically insightful trends were again learned. For example, three types of melodic direction were inferred: primarily upwards (component 1), downwards (3), and relatively stationary (2). In intervallic terms, zero intervals were either very common (components 2 and 3) or they were replaced by a larger dispersion among the remaining intervals (1). While syncopation and/or repeated notes are likely causes for the first trend, Parker's common use of chromatic eighths is likely cause for the second.

[4] We use component and cluster synonymously.

[5] Chorus 5, bar 8 is blank; Parker rested.



(a) PC Clustering (b) Conglomerate Clustering



(c) PC Comp (d) INT Comp (e) DIR Comp

Contexts C, D, E and F

Figure 3: Learning Results

By combining the individual abstract clusters into a more sensitive *conglomerate clustering* – i.e., $< PC\ comp, INT\ comp, DIR\ comp >$ – one arrives at the viewpoint in Figure 3(b). There are 24 distinct clusters in this figure, the most common accounting for 12 of the bars that Parker played (large black square; e.g., contexts C-F: chorus 2, bars 6,7; chorus 4, bar 11; and chorus 10, bar 10, respectively). *The important point is how different contexts C-F appear on the surface* – one would not expect such an abstraction to emerge with a string-of-notes approach. And yet this clustering, comprised of components $< 4, 2, 2 >$, equates these bars because the B♭ Bebop Dom scale is prominent, larger intervals (including zero) occur, and melodic direction is relatively evenly distributed. Similarly, bars 2 and 3 of chorus 5 (Contexts A and B) were also clustered together (large gray square with diamond). Their equivalence, based on components $< 2, 3, 3 >$, relies on the E♭ Maj scale, smaller intervals (including zero), and downward melodic direction.

3 Conclusion

In this paper, we introduce BoB, a model of improvisation that enables the computer to trade solos with a musician in an adaptive, user-specific manner. In pursuing this endeavor, we developed unsupervised learning methods for autonomously customizing the model via improvised examples and demonstrated the powerful musical abstractions that emerge when applied to Charlie Parker's *Mohawk* improvisations. Our key technical contribution was to develop an architecture that naturally enables unsupervised learned knowledge, perception and generation to be tightly coupled.

The granularity of BoB's musical perception depends on a variety of factors – what abstract feature sets are used, how many components are chosen, histogram locality, etc. As one uses more components (and/or combines more feature sets), perception becomes more fine-grained – equivalent bars become more homogeneous. At the same time, learning in the presence of more components/features requires more training data in order to avoid overfitting the model. In addition, as histograms become more local, they contain fewer events and the accuracy of multinomial component discrimination degrades.

Improvisationally speaking, investigating these representational issues involves the following tradeoffs:

1. How homo/heterogeneous must our abstractions be? The tradeoff is between generalization, which affects BoB's ability to produce musically interesting material, and predictability, which relates to responsiveness.
2. Should we use more training data, e.g., combine multiple songs? Just how much of an improvisation's salience is song-dependent?
3. How local should note sequences be? While bars are less musically compelling than phrases, they do have an advantage. *If* one can learn musically salient abstractions with a bar-basis, then the system can ignore the difficult issue of phrase detection. Also note that our choice of bar is arbitrary; these techniques can apply to any VLT basis.

In addressing these issues, we are currently experimenting with various instantiations of BoB in the interactive solo-practice setting.

Acknowledgments Thanks to Manuela Veloso and Roger Dannenberg for their contributions and to the Computer Science Dept. for their support.

References

[1] *Charlie Parker Omnibook: For C Instruments.* Michael H Goldsen, Atlantic Music Corp., 1978.

[2] R. Dannenberg and B. Mont-Reynaud. Following an improvisation in real time. In *Proceedings of the ICMC*, 1987.

[3] PG Music Inc. Band-in-a-box. Technical report, http://www.pgmusic.com, 1999.

[4] P.N. Johnson-Laird. Jazz improvisation: A theory at the computational level. In Peter Howell, editor, *Representing Musical Structure*, pages 291–325. Academic Press, Inc., 1991.

[5] Carol M. Krumhansl. *Cognitive Foundations of Musical Pitch.* Oxford University Press, 1990.

[6] B. Pennycook and D. Stammen. Real-time recognition of melodic fragments using the dynamic timewarp algorithm. In *Proceedings of the ICMC*, 1993.

[7] P. Rolland and J. Ganascia. Automated motive-oriented analysis of musical corpuses: a jazz case study. In *Proceedings of the ICMC*, 1996.

[8] Belinda Thom. Interactive, customized generation of jazz improvisation: Believable music companions. Thesis proposal, unpublished, 1997.

[9] Belinda Thom. Clustering variable-sized samples from discrete nominal distributions. Technical report, Computer Science Department, Carnegie Mellon University, 1999. In progress.