Control Improvisation With Application to Music

Alexandre Donzé[1], Sophie Libkind[2], Sanjit Seshia[1], and David Wessel[1]

University of California, Berkeley

Swarthmore College

September 29, 2013

Control Improvisation Problem

Generic Goal:

- Generate control inputs in a changing environment
- "improvised" $\implies$ randomized
- Soft constraint: stay "close" to some nominal controller
- Hard constraints: always satisfy some specifications

In the musical context:

- Generate random melodies in a given harmonic context
- Generated melodies that "sound similar" to a reference melody

Control Improvisation Problem

Generic Goal:

- Generate control inputs in a changing environment
- "improvised" $\implies$ randomized
- Soft constraint: stay "close" to some nominal controller
- Hard constraints: always satisfy some specifications

In the musical context:

- Generate random melodies in a given harmonic context
- Generated melodies that "sound similar" to a reference melody

Control Improvisation Problem

Formally, given

- A specification FSM $\mathcal{A}^s$,
- A reference accepting word $w_{\textsf{ref}}$,
- A similarity (or "creativity") measure d between words,
- A target distance (or "creativity") d_r from $w_{\textsf{ref}}$
- Any $\epsilon, \delta > 0$

Find a FSA $\mathcal{A}^c$ such that

$$w \in \mathcal{L}(\mathcal{A}^s || \mathcal{A}^c) \implies Pr(|d(w, w_{\textsf{ref}}) - d_r| < \delta) > 1 - \epsilon$$

The target creativity controls how far the improvisation should be from $w_{\textsf{ref}}$

Solving the Control Improvisation Problem

General Idea

- Use learning to create a generator automaton $\mathcal{A}_g$ from w_{ref}
- Use supervisory control $\mathcal{A}_{sc}$ of the generator to enforce the specification, so that $\mathcal{A}_c = \mathcal{A}_g || \mathcal{A}_{sc}$
- Tune probabilistic transitions to meet the creativity criterion

Factor Oracle-based improvisations

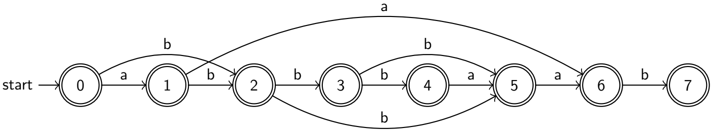
1. Input: a training melody w, represented as a finite sequence of notes, e.g., `abbbaab`
2. Learning: construct a compact structure (factor oracle) representing sub-sequences of w, e.g.,

3. Generation: assign probabilities to transitions of the factor oracle and loop on repeated suffixes

Factor Oracle-based improvisations

1. Input: a training melody w, represented as a finite sequence of notes, e.g., `abbbaab`

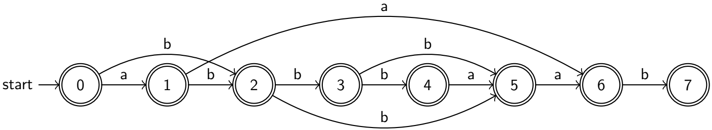
2. Learning: construct a compact structure (factor oracle) representing sub-sequences of w, e.g.,



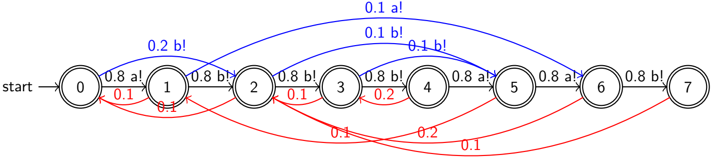
3. Generation: assign probabilities to transitions of the factor oracle and loop on repeated suffixes

Factor Oracle-based improvisations

1. Input: a training melody w, represented as a finite sequence of notes, e.g., `abbbaab`
2. Learning: construct a compact structure (factor oracle) representing sub-sequences of w, e.g.,



3. Generation: assign probabilities to transitions of the factor oracle and loop on repeated suffixes



Properties of Factor Oracles [Assayag and Dubnov, 2004]

A FO reproduces factors of the original string with high probabilities.

With low probabilities, it branches away, generating creativity.

Advantage

Unsupervised learning: no structure is assumed on the input

Disadvantage

Branching likely breaks the underlying structure of the original melody and creates violations of musical specifications

Musical Specifications

It Don't Mean a Thing by Duke Ellington

- **Rhythm specifications:** Jazz melodies can be decomposed into licks seperated by rests. Licks can be forced to begin on certain beats.
- **Pitch specifications:** Notes that sound "weird" with a chord cannot begin a lick and must approach a "good" sounding note.

In practice, we decouple generation of note durations (rhythm) and pitches

Musical Specifications

It Don't Mean a Thing by Duke Ellington

- **Rhythm specifications:** Jazz melodies can be decomposed into licks seperated by rests. Licks can be forced to begin on certain beats.
- **Pitch specifications:** Notes that sound "weird" with a chord cannot begin a lick and must approach a "good" sounding note.

In practice, we decouple generation of note durations (rhythm) and pitches

Musical Specifications

It Don't Mean a Thing by Duke Ellington

- **Rhythm specifications:** Jazz melodies can be decomposed into licks seperated by rests. Licks can be forced to begin on certain beats.
- **Pitch specifications:** Notes that sound "weird" with a chord cannot begin a lick and must approach a "good" sounding note.

In practice, we decouple generation of note durations (rhythm) and pitches

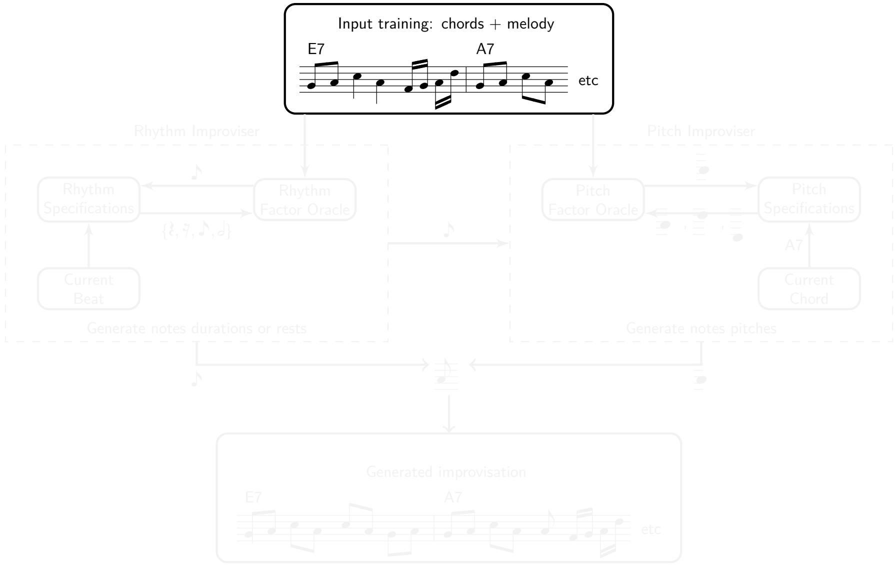
Musical Specifications

It Don't Mean a Thing by Duke Ellington

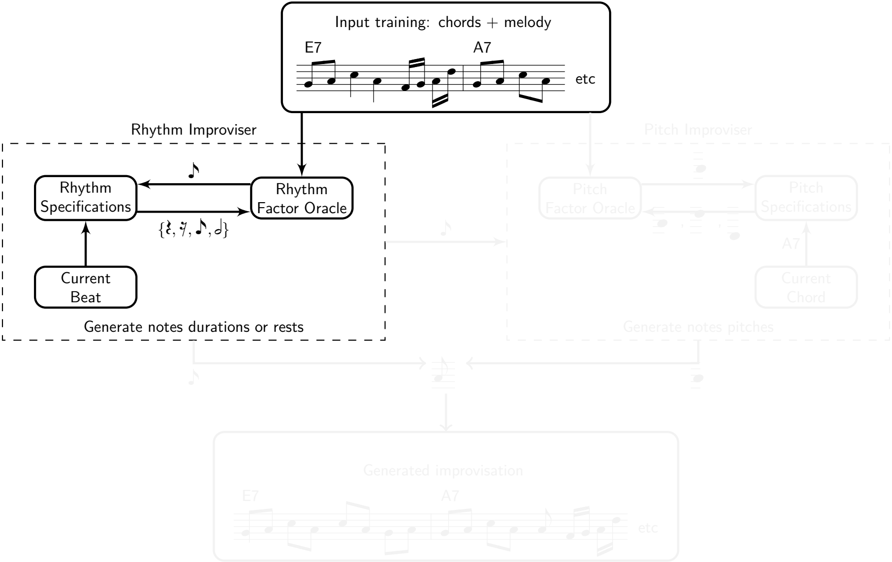
- **Rhythm specifications:** Jazz melodies can be decomposed into licks seperated by rests. Licks can be forced to begin on certain beats.
- **Pitch specifications:** Notes that sound "weird" with a chord cannot begin a lick and must approach a "good" sounding note.

In practice, we decouple generation of note durations (rhythm) and pitches

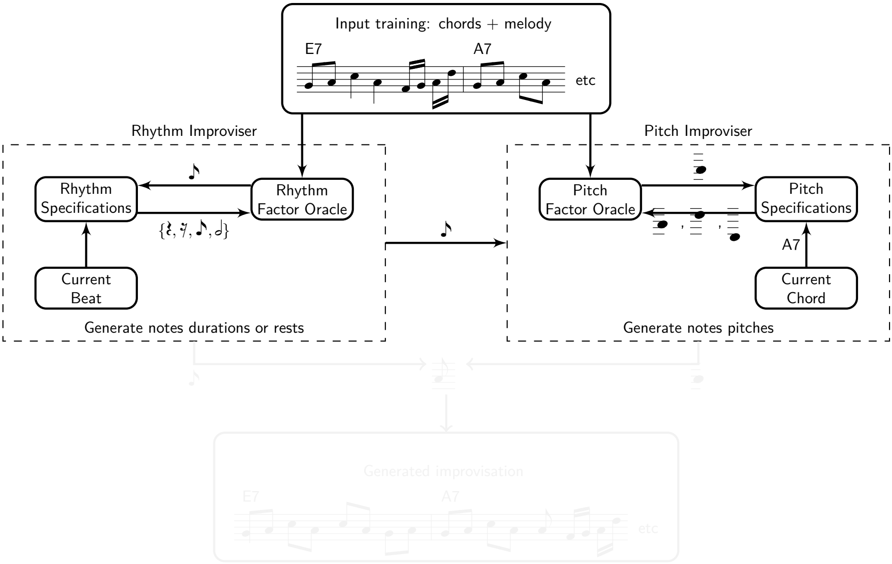
Improviser


Input training: chords + melody
E7
A7
etc
Rhythm Improviser
Rhythm Specifications
Rhythm Factor Oracle
Current Beat
Generate notes durations or rests
Pitch Improviser
Pitch Factor Oracle
Pitch Specifications
A7
Current Chord
Generate notes pitches
Generated improvisation
E7
A7
etc

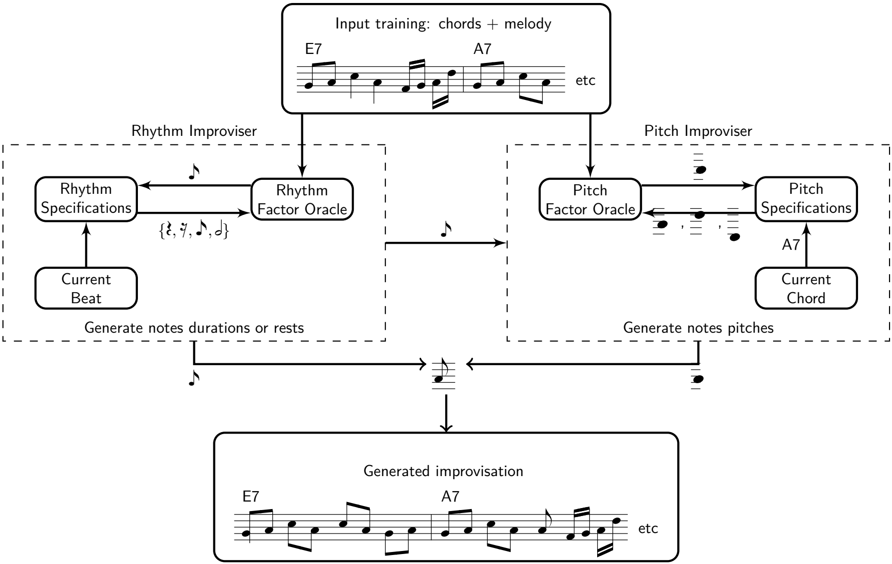
Improviser


Input training: chords + melody
E7
A7
etc
Rhythm Improviser
Rhythm Specifications
Rhythm Factor Oracle
Current Beat
Generate notes durations or rests
Pitch Improviser
Pitch Factor Oracle
Pitch Specifications
A7
Current Chord
Generate notes pitches
Generated improvisation
E7
A7
etc

Improviser


Input training: chords + melody
E7
A7
etc
Rhythm Improviser
Pitch Improviser
Rhythm Specifications
Rhythm Factor Oracle
Pitch Factor Oracle
Pitch Specifications
Current Beat
Current Chord
A7
Generate notes durations or rests
Generate notes pitches
Generated improvisation
E7
A7
etc

Improviser



Input training: chords + melody
E7
A7
etc
Rhythm Improviser
Rhythm Specifications
Rhythm Factor Oracle
Current Beat
Generate notes durations or rests
Pitch Improviser
Pitch Factor Oracle
Pitch Specifications
A7
Current Chord
Generate notes pitches
Generated improvisation
E7
A7
etc

Measuring Creativity

Two sources of divergence:

- Replication probability (probability of not branching in the FO)
- To avoid deadlocks (no note proposed by the generator satisfy specs), we allow to pick notes not proposed by the FO

To measure divergence, we use distances based on compression algorithms.

Compression and Complexity

$K(y)$ is the Kolmogorov complexity of an object y is the length of the shortest compressed code to which it can be losslessly reduced.

We define a measure of creativity which is a variant of the Normalized Compression Distance:

Creativity Divergence (CD)

Approximates amount of information in y not in x.

$$CD(x, y) = \frac{K(y|x)}{K(y)}.$$

$K(y) \approx C(y)$ where

$$C(y) = \text{length of } compress(y)$$

for a compression algorithm $compress$.

Compression and Complexity

$K(y)$ is the Kolmogorov complexity of an object y is the length of the shortest compressed code to which it can be losslessly reduced.

We define a measure of creativity which is a variant of the Normalized Compression Distance:

Creativity Divergence (CD)

Approximates amount of information in y not in x.

$$CD(x, y) = \frac{K(y|x)}{K(y)}.$$

$K(y) \approx C(y)$ where

$$C(y) = \text{length of } compress(y)$$

for a compression algorithm $compress$.

Compression and Complexity

$K(y)$ is the Kolmogorov complexity of an object y is the length of the shortest compressed code to which it can be losslessly reduced.

We define a measure of creativity which is a variant of the Normalized Compression Distance:

Creativity Divergence (CD)

Approximates amount of information in y not in x.

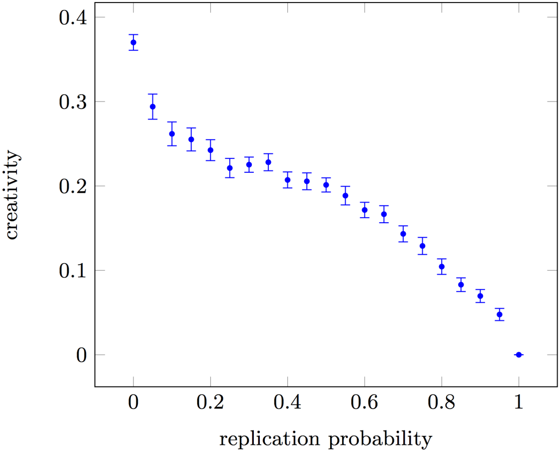
$$CD(x, y) = \frac{K(y|x)}{K(y)}.$$

$K(y) \approx C(y)$ where

$$C(y) = \text{length of } compress(y)$$

for a compression algorithm $compress$.

Evaluating Improvisations



Average creativity with respect to rhythm of 100 improvisations generated by supervise factor oracle.

Controlling Similarity via Compression-based Improvisation

Why?

- Dictionary based compression relies on pattern finding/matching.
- We can control creativity by controlling the difference between $C(improv)$ and $C(orig + improv)$.

Idea

Given an original training melody $orig$ and its compressed form we can synthesize an improvisation $improv$ such that

- $improv$ meets specifications
- $CD(orig, improv)$ is in a specified range

Controlling Similarity via Compression-based Improvisation

Why?

- Dictionary based compression relies on pattern finding/matching.
- We can control creativity by controlling the difference between $C(improv)$ and $C(orig + improv)$.

Idea

Given an original training melody $orig$ and its compressed form we can synthesize an improvisation $improv$ such that

- $improv$ meets specifications
- $CD(orig, improv)$ is in a specified range

Conclusion and Future Work

Summary

- We implemented a supervised musical improviser based on Factor oracles
- We proposed a divergence "creativity" measure
- We derived a new improvisation scheme controlling creativity divergence

Future work

- Further experiments with listeners exposed to our creativity measure
- Real-time (current work with Max/MSP and Ptolemy via OSC)
- Application of control improvisation to other domains

Conclusion and Future Work

Summary

- We implemented a supervised musical improviser based on Factor oracles
- We proposed a divergence “creativity” measure
- We derived a new improvisation scheme controlling creativity divergence

Future work

- Further experiments with listeners exposed to our creativity measure
- Real-time (current work with Max/MSP and Ptolemy via OSC)
- Application of control improvisation to other domains