MUME 2016 - The Fourth International Workshop on Musical Metacreation, ISBN #978-0-86491-397-5

MUS-ROVER: A Self-Learning System for Musical Compositional Rules

Haizi Yu[1] **Lav R. Varshney**[2] **Guy E. Garnett**[3] **Ranjitha Kumar**[1]
[1]Department of Computer Science [2] Department of Electrical and Computer Engineering [3] School of Music
University of Illinois at Urbana-Champaign, Urbana, IL 61801
{haiziyu7,varshney,garnett,ranjitha}@illinois.edu

Abstract

Throughout music history, theorists have identified and documented rules that capture the decisions of composers. This paper asks, "Can a machine behave like a music theorist?" It presents MUS-ROVER, a self-learning system for automatically discovering rules from symbolic music. MUS-ROVER performs feature learning via n-gram models to extract compositional rules — statistical patterns over the resulting features. We evaluate MUS-ROVER on Bach's (SATB) chorales, demonstrating that it can recover known rules, as well as identify new, characteristic patterns for further study. We discuss how the extracted rules can be used in both machine and human composition.

1 Introduction

For centuries, music theorists have developed concepts and rules to describe the regularity in music compositions. Pedagogues have documented commonly agreed upon compositional rules into textbooks (*e.g.*, Gradus ad Parnassum) to teach composition. With recent advances in artificial intelligence, computer scientists have translated these rules into programs that automatically generate different styles of music (Cope 1996; Biles 1994). However, this paper studies the *reverse* of this pedagogical process, and poses the question: can a machine independently extract from symbolic music data compositional rules that are instructive to both machines and humans?

This paper presents MUS-ROVER, a self-learning system for discovering compositional rules from raw music data (*i.e.*, pitches and their durations). Its rule-learning process is implemented through an iterative loop between a *generative* model — "the student" — that emulates the input's musical style by satisfying a set of learned rules, and a *discriminative* model — "the teacher" — that proposes additional rules to guide the student closer to the target style. The self-learning loop produces a rule book and a set of reading instructions that are customized for different types of users.

MUS-ROVER is currently designed to extract rules from four-part music performed by single-line instruments. We represent compositional rules as probability distributions over features abstracted from the raw music data. MUS-ROVER leverages an evolving series of n-gram models over these higher-level feature spaces to capture potential rules from both horizontal and vertical dimensions of the texture.

To evaluate MUS-ROVER, we train the system on Bach's (SATB) chorales (transposed to C), which have been an attractive corpus for analyzing knowledge of voice leading, counterpoint, and tonality due to their relative uniformity of rhythm (Taube 1999; Rohrmeier and Cross 2008). We demonstrate that MUS-ROVER is able to automatically recover compositional rules for these chorales that have been previously identified by music theorists. In addition, we present new, human-interpretable rules discovered by MUS-ROVER that are characteristic of Bach's chorales. Finally, we discuss how the extracted rules can be used in both machine and human composition.

2 Related Work

Researchers have built expert systems for automatically analyzing and generating music. Many analyzers leverage predefined concepts (*e.g.*, chord, inversion, functionality) to annotate music parameters in a pedagogical process (Taube 1999), or statistically measure a genre's accordance with standard music theory (Rohrmeier and Cross 2008). Similarly, automatic song writers such as EMI (Cope 1996) and GenJem (Biles 1994) rely on explicit, ad-hoc coding of known rules to generate new compositions (Merz 2014).

In contrast, other systems generate music by learning statistical models such as HMMs and neural networks that capture domain knowledge — patterns — from data (Simon, Morris, and Basu 2008; Mozer 1994). Recent advances in deep learning take a step further, enabling knowledge discovery via feature learning directly from raw data (Bengio 2009; Bengio, Courville, and Vincent 2013; Rajanna et al. 2015). However, the learned, high-level features are implicit and non-symbolic with post-hoc interpretations, and often not directly comprehensible or evaluable.

MUS-ROVER both automatically extracts rules from raw data — without prior encoding any domain knowledge — and ensures that the rules are interpretable by humans. Interpretable machine learning has studied systems with similar goals in other domains (Malioutov and Varshney 2013; Dash, Malioutov, and Varshney 2015).

3 MUS-ROVER Overview

MUS-ROVER extracts compositional rules — probability distributions over learned features — from both horizontal and vertical dimensions of the texture. MUS-ROVER prioritizes vertical rule extractions via a *self-learning loop*, and learns horizontal rules through a series of *evolving n-grams*.

3.1 Self-Learning Loop

The self-learning loop identifies vertical rules about *sonority* (a *chord* in traditional harmonies) constructions. Its two main components are "the student" — a *generative* model that *applies* rules, and "the teacher" — a *discriminative* model that *extracts* rules. The loop is executed iteratively starting with an empty rule set and an unconstrained student who picks pitches uniformly at random. In each iteration, the teacher compares the student's writing style with Bach's works, and extracts a new rule that augments the current rule set. The augmented rule set is then used to retrain the student, and the updated student is sent to the teacher for the next iteration.

This idea of "learning by comparison" and the collaborative setting between a generative and discriminative model are similarly presented in statistical models such as noise-contrastive estimation (Gutmann and Hyvärinen 2010) and generative adversarial networks (Goodfellow et al. 2014). Both models focus on density estimations to approximate the true data distribution for the purpose of generating similar data; in contrast, our methods do explain the underlying mechanisms that generate the data distribution, such as the compositional rules that produce Bach's styles.

3.2 Evolving n-grams on Feature Spaces

MUS-ROVER employs a series of n-gram models (with *words* being vertical features) to extract horizontal rules that govern the transitions of the sonority features.[1] All n-grams encapsulate copies of self-learning loops to accomplish rule extractions in their contexts. Starting with unigram, MUS-ROVER gradually evolves to higher order n-grams by initializing an n-gram student from the latest (n-1)-gram student. While the unigram model only captures vertical rules such as concepts of intervals and triads, the bigram model searches for rules about sonority progressions such as parallel/contrary motions. This paper only discusses unigram and bigram models; however, higher order n-grams with wider horizontal visions can be trained in a similar manner.

MUS-ROVER's n-gram models operate on high-level feature spaces, which is in stark contrast with many other n-gram applications in which the words are the raw inputs. In other words, a higher-order n-gram in MUS-ROVER shows how vertical features (high-level abstractions) transition horizontally, as opposed to how a specific chord is followed by other chords (low-level details). Therefore, MUS-ROVER does not suffer from low-level variations in the raw inputs, highlighting a greater generalizability.

[1] It potentially learns something about the rhythms, since the transition model probabilistically tells whether to stay or to alter.

4 Unigram MUS-ROVER

Unigram MUS-ROVER and its self-learning loop treat every individual sonority independently of their preceding one(s). It learns the basics of single sonority construction in a choral piece. Restricted by the independence assumption, horizontal rules that describe sonority progression can only be captured in an n-gram for $n \geq 2$ (Section 5).

4.1 Musical Raw Representation

Given a choral piece, we analyze the *symbolic* representation of sheet music rather than a waveform representation of audio. We only leverage the raw representation of the piece — pitches and their durations — and no other higher-level information (*e.g.*, key/time signatures, modes, dynamics). Hence, every choral piece is treated as multiple simultaneously emitting sequences of (discrete) pitch symbols.

We use integer-valued *MIDI numbers* as opposed to letter names, so we can perform arithmetic operations such as calculating the (piano) distance between two pitches in the unit of semitones by subtraction. We restrict attention to a finite set of MIDI numbers $\Omega = \{21, 22, \ldots, 108\}$, establishing a bijection to the 88 piano keys: 21/108 to the leftmost/rightmost key (A0/C8) and 60 to middle C (C4).

We represent a four-part chorale as a four-row matrix $X \in \Omega^{4\times N}$, whose entries are MIDI numbers. The rows represent the horizontal *melodies* in each voice: the 1st, 2nd, 3rd, and 4th row of X correspond to soprano, alto, tenor, and bass, respectively. The columns represent the vertical four-pitch sonorities, where each column has unit duration equaling the greatest common divisor (gcd) of note durations in the piece. For instance, if the choral piece is composed from quarter notes and dotted quarter notes only (the gcd of these two types of notes is an eighth note), then a quarter note C4 will result in a sequence of repeated 60s spanning 2 consecutive columns (3 columns for a dotted quarter note). The MIDI matrix is all that is needed in the sequel, which is referred as the piece's *raw* representation.

4.2 Student: the Generative Model

The student is a *probabilistic composer*, which generates chorales by sampling from a probability distribution. Its composition gets probabilistically closer to Bach's style via more Bach-like distributions that are computed from applying compositional rules. The I/O of the student:

Input: a set of compositional rules;
Output: a probabilistic model.

4.2.1 Probabilistic Assumption We refer to the *probabilistic model* of the student as the *choral distribution* P of chorales in terms of MIDI matrices. This assumes that the student composes every choral piece as a random sample from P. Under an n-gram model, P factorizes as the product of the (conditional) *sonority distributions*. Thus, learning the student's probabilistic model boils down to learning the (conditional) sonority distributions.

To formalize in the unigram setting, consider the MIDI matrix as a random variable $\mathbf{X} = [\mathbf{X}_{\bullet 1}, \cdots, \mathbf{X}_{\bullet N}] \in \Omega^{4\times N}$, where $\mathbf{X}_{\bullet i}$ is the ith column of $\mathbf{X}$, and N denotes

the length of the piece. Under the independence assumption of unigram, the choral distribution factorizes as the product of sonority distributions:

$$\mathsf{P}(\mathbf{X}) = \prod_{i=1}^{N} p\left(\mathbf{X}_{\bullet i}\right).$$

By stationarity, all sonorities follow the same probability mass function (pmf) $p : \Omega^4 \mapsto [0,1]$. Unlike the infinite sample space of P, the sample space of p is discrete and finite, with size $d = \left|\Omega^4\right| = 88^4$. If we impose an ordering in its sample space, p can be represented as a vector that lies in the d-dimensional probability simplex,[2] Δ^d.

4.2.2 Learning Sonority Distribution (p) The sonority distribution p is computed from a set of compositional rules. To constrain the student by the current rules while also providing encouragement to try all feasible sonorities with equal probability, the following optimization is performed in the kth iteration:

$$\begin{aligned} \underset{p\in\Delta^d}{\text{maximize}} \quad & I(p) \\ \text{subject to} \quad & p \in \Gamma_1, \ldots, p \in \Gamma_k. \end{aligned} \tag{1}$$

The ith constraint $p \in \Gamma_i$ requires p to satisfy the ith rule. Formalization of composition rules and corresponding constraints are detailed in Sections 4.3.1 and 4.3.5, respectively. The objective function $I : \Delta^d \mapsto \mathbb{R}$ is a Tsallis entropy, which achieves a maximum when p is uniform and a minimum when p is deterministic. Thus the constrained maximization of $I(p)$ disperses probability mass across all feasible possibilities and thereby encourages creativity from randomness. Common choices of Tsallis entropy $I(p)$ are the Gini impurity $I_G(p) = 1 - \sum_i p_i^2$, and Shannon entropy $I_E(p) = -\sum_i p_i \log p_i$.

4.2.3 Implementation as Linear Least-Squares For computational efficiency, we prefer Gini impurity as the objective function, so maximizing $I_G(p)$ is equivalent to minimizing $\|p\|_2^2$. Later in Section 4.3.5, (9) will show that each constraint $p \in \Gamma_i$ is a set of linear equality constraints $A^{(i)}p = b^{(i)}$, where $A^{(i)}$ and $b^{(i)}$ are derived from the ith rule. If we further introduce $A^{(0)} = [1, 1, \cdots, 1]$ and $b^{(0)} = 1$, and stack $\{A^{(i)}\}_{i=0}^k, \{b^{(i)}\}_{i=0}^k$ together to form

$$A = \begin{bmatrix} A^{(0)} \\ A^{(1)} \\ \vdots \\ A^{(k)} \end{bmatrix}, \quad b = \begin{bmatrix} b^{(0)} \\ b^{(1)} \\ \vdots \\ b^{(k)} \end{bmatrix},$$

optimization problem (1) can be rewritten as a quadratic program (QP):

$$\begin{aligned} \text{minimize} \quad & \|p\|_2^2 \\ \text{subject to} \quad & Ap = b, \\ & 0 \preceq p \preceq 1. \end{aligned} \tag{2}$$

[2] $\Delta^d = \left\{ p \in [0,1]^d \;\middle|\; \sum_{i=1}^d p_i = 1 \right\}$. We overload the notation p to denote both the pmf and its vector representation. Its actual meaning can be readily figured out from the context.

Rules may conflict, and so equality constraints may only hold approximately: $Ap \approx b$. Therefore, the magnitude of the residual $\|Ap - b\|_2^2 = \sum_i \|A^{(i)}p - b^{(i)}\|_2^2$ measures the efficacies of the rules and reflects compromises among them. Taking this approximation into consideration yields the following constrained linear least-squares problem:

$$\begin{aligned} \text{minimize} \quad & \|Ap - b\|_2^2 + \lambda\|p\|_2^2 \\ \text{subject to} \quad & 0 \preceq p \preceq 1, \end{aligned} \tag{3}$$

where $\lambda > 0$ balances the trade-off between a progressive ($\lambda \to \infty$) and a conservative ($\lambda \to 0$) student. As a final step, we normalize the solution of (3): $p^\star / \|p^\star\|_1$, so as to make it a valid pmf — the learned probabilistic model.

4.3 Teacher: the Discriminative Model

The teacher is a *probabilistic connoisseur*, which induces one feature per iteration whose probability distribution best discriminates the student from Bach. The feature serves as a high-level abstraction of the piece, which provides one angle to analyze music. A compositional rule is then derived by computing the empirical feature distribution from Bach's chorales. The I/O of the teacher:

Input 1: a collection of Bach's chorales (positive data),
Input 2: a probabilistic student[3] (negative data);
Output: one additional compositional rule.

4.3.1 Compositional Rule (ϕ, p_ϕ) A *compositional rule* is defined by a feature and its probability distribution, which reveals certain levels of regularity/speciality that is otherwise buried in the raw representation of the music. In the unigram setting, since the student's probabilistic model is specified by the sonority distribution, we study features of vertical sonorities in particular. To formalize, let $\mathbf{x} \in \Omega^4$ be a random (sonority) vector, and introduce a feature map ϕ and the corresponding feature distribution p_ϕ:

$$\phi : \Omega^4 \mapsto \phi(\Omega^4), \qquad p_\phi : \phi(\Omega^4) \mapsto [0,1],$$

where ϕ is a deterministic function that maps the raw representation of a sonority to some feature space, and p_ϕ is a pmf of $\phi(\mathbf{x})$. A compositional rule is the pair $r = (\phi, p_\phi)$. The next two sub-sections elaborate a rule's two components: ϕ and p_ϕ, respectively.

4.3.2 Feature Map (ϕ) Construction We introduce an automatic procedure to construct a rich class of feature maps that are simple and semantically meaningful. Such automation is initialized from a small set of basis features, named *descriptors*, and a large set of selectors, named *windows*. It constructs feature maps by combinatorially enumerating the descriptors and windows.

To formalize, we define a feature map ϕ as the composition of a descriptor d and a selection window w. Given a set D of descriptors and a set W of windows, MUS-ROVER's feature constructor generates a rich class of feature maps

$$\Phi = \{d \circ w \mid d \in D, w \in W\}. \tag{4}$$

[3] Alternatively, one can provide another choral dataset written by a human student who iterates with MUS-ROVER for feedback.

In (4), a selection *window* w is a function that maps a sonority $x \in \Omega^4$ to a partial sonority by selecting only the designated part(s). For instance, if we use a subscript to denote the selected part(s), then $w_{\{1,4\}}(x) = (x_1, x_4)$ picks the soprano and bass pitches. A *descriptor* d is a function whose domain is the space of partial sonorities.

There is a stark contrast in constructing W and D. W is a large set that contains all types of vertical windows of size ≤ 4, *i.e.*, $W = \left\{w_I \;\middle|\; I \in 2^{\{1,2,3,4\}} \backslash \{\emptyset\}\right\}$, thus is easily enumerable. On the contrary, D is hand-designed. However, unlike most hand designs, which require much domain knowledge and intuition, the construction of D is easy. This is realized by demanding 1) $|D|$ is small, and 2) $\forall d \in D$, d is a basic arithmetic operation that requires no music expertise, but only basic observations from an ordinary person. In this paper, we only leverage the periodicity and distance measures of a piano keyboard, and choose $D = \{d_{pitch}, d_{pitch12}, d_{interv}, d_{interv12}, d_{order}\}$ where

$$d_{pitch}(z) = z,\ \forall z \in \Omega \cup \Omega^2 \cup \Omega^3 \cup \Omega^4;$$
$$d_{pitch12}(z) = \text{mod}(z, 12),\ \forall z \in \Omega \cup \Omega^2 \cup \Omega^3 \cup \Omega^4;$$
$$d_{interv}(z) = \text{abs}(\text{diff}(z)),\ \forall z \in \Omega^2 \cup \Omega^3 \cup \Omega^4;$$
$$d_{interv12}(z) = \text{mod}(\text{abs}(\text{diff}(z)), 12),\ \forall z \in \Omega^2 \cup \Omega^3 \cup \Omega^4;$$

and $d_{order}(z)$ maps $z \in \Omega^2 \cup \Omega^3 \cup \Omega^4$ to a string that specifies the ordering of its elements, *e.g.*, $d_{order}((60, 55, 52, 52)) =$ "4=3<2<1". The numbers in an order string denote the indices of the input vector.

We use an example to walk through the set of descriptors: for a partial sonority $z = (72, 52, 55)$, $d_{pitch}(z) = (72, 52, 55)$, $d_{pitch12}(z) = (0, 4, 7)$, $d_{interv}(z) = (20, 3)$, $d_{interv12}(z) = (8, 3)$, and $d_{order}(z) =$ "2<3<1".

Constructing features from (4) is simple and semantically meaningful as desired. For instance, people can simply read out the feature specified by $d_{interv} \circ w_{\{1,4\}}$ as the piano distance between the soprano and bass pitches.

4.3.3 Feature Distribution (p_ϕ) Inference We introduce two inference procedures — exact and approximate — depending on whether we examine a composer by its probabilistic model or by its works.

For a probabilistic composer with a known probabilistic model, such as the student, *exact* inference is possible. In the unigram setting, the probabilistic model is the sonority distribution $p(\mathbf{x})$ of the raw representation $\mathbf{x} \in \Omega^4$. Hence, given a feature $\mathbf{y} = \phi(\mathbf{x})$, the pmf of $\mathbf{y}$ is computed as

$$p_\phi(y) = \sum_{x \in \phi^{-1}(\{y\})} p(x). \tag{5}$$

For a non-probabilistic individual, such as Bach, we perform approximate inference to estimate the feature distribution from his/her works. In the unigram setting, given a random choral piece $\mathbf{X}$ as a random MIDI matrix, $\phi(\mathbf{X}_{\bullet 1}), \ldots, \phi(\mathbf{X}_{\bullet N}) \overset{i.i.d.}{\sim} p_\phi$. Hence, we derive the *maximum likelihood estimation* (MLE) of p_ϕ by its empirical distribution $\hat{p}_{\phi|\mathcal{C}}$ from a choral collection $\mathcal{C}$

$$p_\phi(y) \approx \hat{p}_{\phi|\mathcal{C}}(y) = \frac{1}{Z} q_{\phi|\mathcal{C}}(y) = \frac{1}{Z} \sum_{x \in \mathcal{C}} \mathbb{1}[\phi(x) = y], \tag{6}$$

where $q_{\phi|\cdot}$ represents the feature counts given data, and $Z = \sum_{y \in \phi(\Omega^4)} q_{\phi|\mathcal{C}}(y)$ is the normalizing constant.

To summarize, we have thus far introduced a compositional rule, a pair (ϕ, p_ϕ), in which ϕ is a feature map obtained from (4), and p_ϕ is the feature distribution obtained from (5) or (6). In the sequel, Section 4.3.4 and 4.3.5 will talk about extracting rules and post-processing rules as constraints, respectively.

4.3.4 Automatic Rule Extraction In each iteration, the teacher scrutinizes a list of unlearned features, and filters out one that satisfies certain desired properties. The chosen feature and its distribution form the rule in the current iteration. We name and formalize the desired properties of a rule (ϕ, p_ϕ), so that it is worth being extracted:

1. The feature distribution p_ϕ is intended to be *discriminative*, so that it discloses a large gap between the student and Bach. The Kullback–Leibler (KL) divergence is used to quantify such a gap, *a.k.a.*, Bayesian surprise for the rule (Varshney 2013).
2. The feature distribution p_ϕ from Bach is intended to be *informative*, so that it shows regularity in Bach's chorales and is easy for humans to remember and apply. The Shannon entropy is used to quantify the level of regularity.
3. The feature ϕ is intended to be a *high-level* representation of a sonority's raw representation. The ratio $\sigma_\phi = |\phi(\Omega^4)|/|\Omega^4|$ is used to quantify the level of abstraction: the smaller σ_ϕ is, the higher level of abstraction ϕ has.
4. The feature ϕ is intended to be *human interpretable*, which requires ϕ to be semantically meaningful.

To formalize the automatic rule extraction process that accounts for the desired properties, we first introduce a few notations to fit in the iterative setting of the self-learning loop. For the kth iteration, let $r^{(k)} = (\phi^{(k)}, p_{\phi^{(k)}})$ be the extracted rule, $R^{\langle k \rangle} = \{r^{(1)}, \cdots, r^{(k)}\}$ be the rule set, and $\Phi^{\langle k \rangle} = \{\phi^{(1)}, \cdots, \phi^{(k)}\}$ be the set of feature maps used in $R^{\langle k \rangle}$. MUS-ROVER starts with $R^{\langle 0 \rangle} = \emptyset$, and learns one additional rule per iteration, so $|R^{\langle k \rangle}| = k, \forall k = 0, 1, 2, \ldots$.

In the kth iteration, the teacher takes as inputs Bach's chorales $\mathcal{C}_{bach}$ and the student's probabilistic model $p^{\langle k-1 \rangle}$ from the previous iteration, and outputs a rule $r^{(k)}$, by solving the following optimization problem for ϕ:

$$\begin{aligned} &\text{maximize} \quad s\left(\hat{p}_{\phi|\mathcal{C}_{bach}},\ p_\phi^{\langle k-1 \rangle}\right) \\ &\text{subject to} \quad \phi \in \Phi \backslash \left(\Phi^{\langle k-1 \rangle} \cup \{\phi_{raw}\}\right). \end{aligned} \tag{7}$$

The objective $s(\cdot, \cdot)$ is a scoring function of two feature distributions: one from Bach, one from the student. In the constraint, Φ is the feature class in (4), and $\phi_{raw} = d_{pitch} \circ w_{\{1,2,3,4\}}$ is a sonority's raw representation.

Among the four desired properties of a rule, the scoring function accounts for the first two — being both discriminative and informative. We draw inspirations from the *cultural hole* (CH) between two distributions, first defined in

networks of scholarly communication (Vilhena et al. 2014): for distributions p, q,

$$CH(p,q) = 1 - \frac{H(p)}{H(p,q)} = \left(1 + \frac{H(p)}{D(p||q)}\right)^{-1},$$

where $H(\cdot)$, $H(\cdot,\cdot)$, and $D(\cdot||\cdot)$ are Shannon entropy, cross entropy, and KL divergence, respectively. CH is grounded in information theory: 1) 1−CH measures similarity: it is the ratio of the average code length needed by Bach to that needed by the student when both are requested to write in Bach's style; 2) CH is asymmetric: it only measures the gap from the student to Bach, as the student is learning from Bach but not vice versa. Maximizing $CH(p,q)$ is equivalent to maximizing the ratio $D(p||q)/H(p)$, which coincides with the first two properties. However, CH does not provide a mechanism to balance the trade-off between maximizing $D(p||q)$ (being discriminative) and minimizing $H(p)$ (being informative). So we generalize CH by introducing a trade-off parameter $\alpha \in [0,1]$ and set the scoring function in (7)

$$s(p,q) = \frac{D(p||q)^{\alpha}}{H(p)^{1-\alpha}}. \tag{8}$$

When $\alpha = 1$, $s(p,q) = D(p||q)$ which ignores the second property. When $\alpha = 0$, $s(p,q) = 1/H(p)$ which ignores the first property. When $0 < \alpha < 1$, both properties are taken into account, and when $\alpha = 0.5$, maximizing the scoring function reduces to maximizing CH.

The last two properties are both realized in the constraint of (7). In the third property, $\sigma_\phi \leq 1, \forall \phi$. As $\sigma_\phi = 1$ implies that ϕ has the same level of abstraction as the raw representation, excluding ϕ_{raw} allows all feasible ϕ of (7) are higher level features ($\sigma_\phi < 1$). The fourth property is realized by the construction of Φ (Section 4.3.2).

Given the solution $\phi^\star$ to (7), the kth compositional rule is $r^{(k)} = (\phi^\star, \hat{p}_{\phi^\star|\mathcal{C}_{bach}})$, which augments the rule set to $R^{\langle k\rangle} = R^{\langle k-1\rangle} \cup \{r^{(k)}\}$.

4.3.5 Rules as Constraints As a last step, the teacher converts a rule $r = (\phi, p_\phi)$ into a constraint $p \in \Gamma$ of optimization problem (1). The conversion is implemented as the converse of the (exact) inference problem for feature distributions (5). To formalize, let $\mathbf{x} \in \Omega^4$ be a random (sonority) vector, then applying the rule (ϕ, p_ϕ) demands that the distribution of $\mathbf{y} = \phi(\mathbf{x})$ is p_ϕ. Solving the sonority distribution p from the feature distribution p_ϕ gives the constraint set

$$\Gamma = \bigcap_{y \in \phi(\Omega^4)} \left\{ p \middle| \sum_{x \in \phi^{-1}(\{y\})} p(x) = p_\phi(y) \right\}, \tag{9}$$

which is a set of linear equality constraints.

5 Bigram MUS-ROVER

Bigram MUS-ROVER is targeted to learn horizontal rules regarding sonority progressions. It studies the transition in various high-level feature spaces that are automatically extracted by the same self-learning loop from its unigram counterpart (with the same set of features Φ). Conditioned on different preceding sonorities, parallel rule-learning is possible via separate copies of the self-learning loop.

5.1 Conditional Self-Learning Loops

The self-learning loop conditioned on a preceding sonority works in a similar manner as in the unigram setting, with differences that are listed in the following sub-sections.

5.1.1 The Probabilistic Model (p_c) The student's probabilistic model follows the bigram setting with a Markov assumption. The choral distribution factorizes as the product of the *conditional* sonority distributions:

$$\mathsf{P}(\mathbf{X}) = \prod_{i=1}^{N} p_c\left(\mathbf{X}_{\bullet i} \mid \mathbf{X}_{\bullet i-1}\right),$$

where $\mathbf{X}_{\bullet 0} = *$, a special character that signifies the start of a piece. By stationarity, all sonorities follow the same conditional distribution $p_c(\cdot|\cdot) : \Omega^4 \times \left(\Omega^4 \cup \{*\}\right) \mapsto [0,1]$, which serves as the student's probabilistic model in the bigram setting. Learning p_c can be parallelized via computing $p_c(\cdot|x_p)$ for every preceding sonority $x_p \in \Omega^4 \cup \{*\}$, which results in separate copies of the self-learning loop. Within each copy associated with x_p, we solve $p_c(\cdot|x_p)$ via optimization problem (1).

5.1.2 Horizontal Rule Extraction A horizontal rule, in the form of $(\phi, p_{\phi|x_p})$, characterizes the transition from the sonority x_p in the scope of the feature ϕ. The inference problem for $p_{\phi|x_p}$ adapts to the bigram setting: for the student, $p_{\phi|x_p}$ is inferred *exactly* via (5) but from the conditional probabilistic model $p_c(\cdot|x_p)$; for Bach, $p_{\phi|x_p}$ is inferred *approximately* as the empirical distribution of $\phi(\mathbf{x})$ given $\phi(x_p)$, written as $\hat{p}_{\phi|x_p,\mathcal{C}_{bach}}$.[4] Then the rule extraction in the bigram setting can also be conducted via optimization problem (7) by replacing $\hat{p}_{\phi|\mathcal{C}_{bach}}$ and $p_\phi^{\langle k-1\rangle}$ with $\hat{p}_{\phi|x_p,\mathcal{C}_{bach}}$ and $p_{\phi|x_p}^{\langle k-1\rangle}$, respectively.

5.2 Online Learning

In principle, one can parallelize as many copies of the self-learning loop as possible to learn the full probabilistic model $p_c(\cdot|\cdot)$ offline. The number equals $|\Omega^4 \cup \{*\}| = 88^4 + 1$, which is too large to be fully parallelized in practice. Instead, we propose an *online* learner — an endless writer — which writes a choral piece as an infinite sequence of sonorities. It starts with $*$, and learns $p_c(\cdot|x_p)$ *on the fly* based on the preceding sonority. The entire online learning process is endless in order to maximize its learning capacity, and is detailed as the following infinite loop:

1) Launch the unigram MUS-ROVER for sufficiently many[5] (say K) iterations, and save the probabilistic model of the student $S^{\langle K\rangle}$ in the final iteration.
2) Start a sonority sequence $\mathcal{S}$ with $*$, and initialize a knowledge base $\mathcal{K} = \emptyset$ for storing probabilistic models.
3) Repeat the following procedures forever:

[4] $\hat{p}_{\phi|x_p,\mathcal{C}_{bach}}$ is *not* the empirical distribution of $\phi(x)$ given x_p. Conditioned on $\phi(x_p)$ rather than x_p eases the pain caused by rare or unseen x_p, and achieves high generalizability.

[5] For instance, when the gap between the student and Bach measured by the (raw) sonority distribution is small enough.

Unigram Rule (ϕ, p_ϕ) Catalog			
Index	*Window*	*Descriptor*	*Entropy*(p_ϕ)
1	(1,4)	order	0.000
2	(1,3)	order	0.006
11	(1,2,3,4)	order	0.691
12	(1,)	pitch12	2.934
16	(1,4)	interv12	3.066
32	(2,3,4)	interv12	5.348
52	(1,2,3,4)	interv12	7.090
63	(1,2,3,4)	pitch	10.091

Table 1: An excerpt of the unigram rule catalog. The complete catalog includes 63 rules sorted by the Shannon entropies of their feature distributions. The window notation uses integers $1, 2, 3, 4$ to denote SATB, respectively.

a) Read the last sonority in $\mathcal{S}$ as x_p.
b) If the probabilistic model $p_c(\cdot|x_p) \in \mathcal{K}$, go to d).
c) If the probabilistic model $p_c(\cdot|x_p) \notin \mathcal{K}$:
   - Launch a self-learning loop in the bigram setting conditioned on x_p, and initialize its student by the unigram student: $S^{\langle 0 \rangle}_{|x_p} = S^{\langle K \rangle}$.
   - Execute the self-learning loop for sufficiently many iterations, and save the final $p_c(\cdot|x_p)$ to the knowledge base: $\mathcal{K} = \mathcal{K} \cup \{p_c(\cdot|x_p)\}$.
d) Sample a sonority from $p_c(\cdot|x_p)$, and append it to the end of $\mathcal{S}$. Go back to a).

It is clear from step 3-c) that a bigram student is evolved from the final unigram student. Such evolution from a lower order n-gram to a higher order, is *key* in rule-learning, since some rules such as "Parallel octaves/fifths are avoided!" are recovered by comparing rules from different n-gram models. We will exemplify this in the following section.

6 Learning Rules from Bach's Chorales

MUS-ROVER outputs two main products: 1) a rule book on Bach's chorales, and 2) customized rule-learning traces for reading the book. The rule book records a lengthy list of rules that summarizes the statistics of Bach's chorales from all possible angles (features) specified by Φ in (4). The traces suggest empirical ways to read this list of conceptually entangling rules — rules that are implied from others.

6.1 A Rule Book on Bach's Chorales

MUS-ROVER independently writes a rule book summarizing the statistical structure of Bach's chorales. Even though the book is authored by a machine, it is designed to be readable by humans. MUS-ROVER *only* leverages its feature constructor to explore probabilistic patterns, regardless of any self-learning loops. Inspecting the rule book allows us to compare the machine-generated rules to our knowledge, and gives us concrete cases from Bach that exemplify the rules. Furthermore, it opens the opportunity for humans to learn music theory from a machine-generated reference.

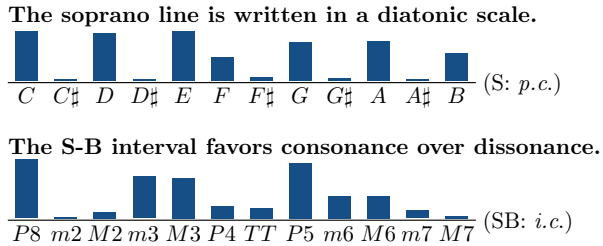


Figure 1: Unigram rule examples from Bach's chorales.

6.1.1 Chapter–1: Unigram Rules The opening chapter records all the unigram rules, whose associated features are automatically generated from (4). All the 63 unigram rules are sorted by the *Shannon entropies* of the feature distributions, a surrogate for human memorability (Pape, Kurtz, and Sayama 2015): rules that are more deterministic are easier for humans to memorize.

Table 1 shows an excerpt of the (sorted) catalog of the unigram rules. The complete catalog delimits the universe of all rules that are reachable via (4), which demonstrates MUS-ROVER's exploration capacity. The first eleven rules in the catalog specify the pitch orderings between/among voices, all of which suggest that the pitch in a higher voice should sound higher. These rules, though less interesting, are consistent with our pedagogical restrictions on voice crossing. The 12th rule considers the soprano voice, and its descriptor $d_{pitch12}$ is semantically equivalent to *pitch class* (*p.c.*). It shows the partition of two *p.c.* sets (Figure 1: top), which says that the soprano line is built on a diatonic scale. The 16th rule considers the soprano and bass, and its descriptor $d_{interv12}$ is semantically equivalent to *interval class* (*i.c.*). It recovers our definition of intervalic quality: consonance versus dissonance (Figure 1: bottom).

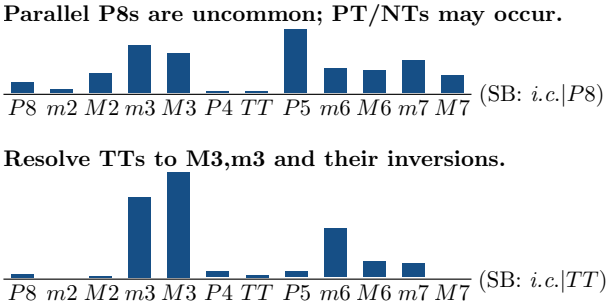
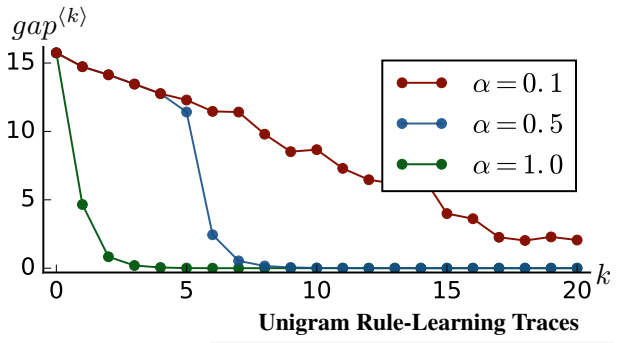


Figure 2: Bigram rule examples from Bach's chorales.

6.1.2 Chapter–2: Bigram Rules The second chapter records the bigram rules that are also summarized from all of the 63 features (Φ). Given a feature ϕ, the bigram rule is represented by the feature transition distribution $p_\phi(\cdot|\cdot)$. Take $\phi = d_{pitch12} \circ w_{\{1,4\}}$ for example: between soprano and bass, $p_\phi(\cdot|*)$ gives the pmf of the opening *i.c.*; $p_\phi(\cdot|7)$ gives the pmf of the *i.c.* after a $P5$. Due to the large amount of conditionals for each feature, Chapter-2 is much longer than Chapter-1.



Unigram Rule-Learning Traces

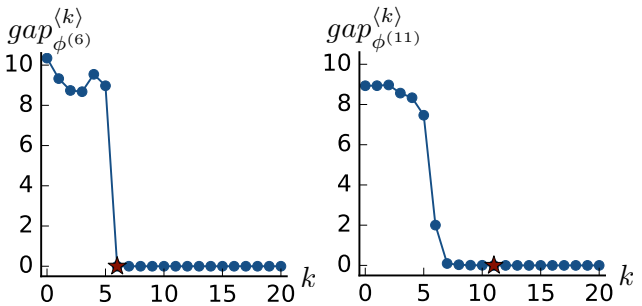
	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1.0$
1	(1,4), order	(1,4), order	(1,2,3), pitch
2	(1,3), order	(1,3), order	(2,3,4), pitch
3	(2,4), order	(2,4), order	(1,2,3,4), pitch12
4	(1,2), order	(1,2), order	(1,3,4), pitch
5	(2,3), order	(2,3,4), order	(1,2,4), pitch
6	(3,4), order	(1,3,4), pitch	(1,2,3,4), interv
...	...	...	...
E_ϵ	∞	12	6
M_ϵ	2.21	4.97	8.63

Table 2: Three unigram rule-learning traces ($\mathcal{T}^{\langle 20 \rangle}$) with $\alpha = 0.1, 0.5, 1.0$. The top figure shows the footprints that mark the diminishing gaps. The bottom table records the first six rules, and shows the tradeoff between efficiency and memorability ($\epsilon = 0.005$). The trace with $\alpha = 1.0$ shows the most efficiency, but the least memorability.

Comparing the top bigram rule in Figure 2 with the bottom unigram rule in Figure 1 shows the re-distribution of the probability mass for feature $d_{pitch12} \circ w_{\{1,4\}}$, the *i.c.* between soprano and bass (SB: *i.c.*). The dramatic drop of $P8$ recovers the rule that avoids parallel P8s, while the rises of $m7, M7$ and their inversions suggest the usage of passing/neighbor tones (PT/NTs). The bottom rule in Figure 2 illustrates *resolution* — an important technique used in tonal harmony — which says tritones (TTs) are most often resolved to $m3, M3$ and their inversions.

Interestingly, the fifth peak ($m7$) in the pmf of the bottom rule in Figure 2 reveals an observation that doesn't fall into the category of (direct) resolution. In one example of $TT \to m7$: $(F4, B2) \to (F4, G2)$ followed by $(E4, C3)$, the transition is similar to the notion of escape tone (ET), which delays the resolution by suspending the tension. In another example of $TT \to m7$: $(F4, B2) \to (G4, A2)$ after $(E4, C\sharp 3)$, the middle sonority behaves as a passing chord between two A7 chords. We hypothesize that, with further study, such transition pattern may potentially lead to some new rule like "voice leading takes precedence over sonority especially when the sonority is on a weak beat and the beats surrounding it are the same harmony (different inversions)".

All of these rules (or their equivalents) are automatically identified during the rule-learning process rather than manually picked, which will be discussed in the sequel.



(a) $\phi^{(6)} = d_{pitch} \circ w_{\{1,3,4\}}$ (b) $\phi^{(11)} = d_{interv} \circ w_{\{1,2,3,4\}}$

Figure 3: Rule entanglement: two sets of footprints that mark the diminishing gaps, both of which are from the rule-learning trace with $\alpha = 0.5$. The location of the star shows whether the associated rule is entangled (right) or not (left).

6.2 Customized Rule-Learning Traces

Despite its readability for every single rule, the rule book is in general hard to read as a whole due to its length and lack of organization. MUS-ROVER's self-learning loop solves both challenges by offering customized *rule-learning traces* — ordered rule sequences — resulting from its iterative extraction. Therefore, MUS-ROVER not only outputs a comprehensive rule book, but more crucially, suggests ways to read/analyze it, tailored to different types of students.

6.2.1 Analyzing Unigram Rules We propose two criteria, *efficiency* and *memorability*, to assess a rule-learning trace from the unigram model. The efficiency measures the speed in approaching Bach's style; the memorability measures the complexity in memorizing the rules. A good trace is both efficient in imitation and easy to memorize.

To formalize these two notions, we first define a rule-learning trace $\mathcal{T}^{\langle k \rangle}$ as the ordered list of the rule set $R^{\langle k \rangle}$, and quantify the gap against Bach by the KL divergence in the raw feature space: $gap^{\langle k \rangle} = D(\, \hat{p}_{\phi_{raw}|\mathcal{C}_{bach}} \parallel p^{\langle k \rangle}_{\phi_{raw}}\,)$. The *efficiency* of $\mathcal{T}^{\langle k \rangle}$ with efficiency level ϵ is defined as the minimum number of iterations that is needed to achieve a student that is ϵ-close to Bach if possible:

$$\mathsf{E}_\epsilon\left(\mathcal{T}^{\langle k \rangle}\right) = \begin{cases} \min\left\{n \;\middle|\; gap^{\langle n \rangle} < \epsilon\right\}, & gap^{\langle k \rangle} < \epsilon; \\ \infty, & gap^{\langle k \rangle} \geq \epsilon. \end{cases}$$

The *memorability* of $\mathcal{T}^{\langle k \rangle}$ is defined as the average entropy of the feature distributions from the first few efficient rules:

$$\mathsf{M}_\epsilon\left(\mathcal{T}^{\langle k \rangle}\right) = \frac{1}{N}\sum_{k=1}^{N} H\left(\hat{p}_{\phi^{(k)}|\mathcal{C}_{bach}}\right),$$

where $N = \min\left\{k, \mathsf{E}_\epsilon\left(\mathcal{T}^{\langle k \rangle}\right)\right\}$. There is a tradeoff between efficiency and memorability. At one extreme, it is most efficient to just memorize $p_{\phi_{raw}}$, which takes only one step to achieve a zero gap, but is too complicated to memorize or learn. At the other extreme, it is easiest to just memorize p_ϕ for ordering related features, which are (nearly) deterministic but less useful, since memorizing the orderings

takes you acoustically nowhere closer to Bach. The α parameter in the scoring function of (7) is specially designed to balance the tradeoff, with a smaller α for more memorability and a larger α for more efficiency (Table 2).

To study the rule entangling problem, we generalize the notion of gap from the raw feature to all high-level features:

$$gap_{\phi}^{\langle k \rangle} = D(\, \hat{p}_{\phi|\mathcal{C}_{bach}} \parallel p_{\phi}^{\langle k \rangle} \,), \quad \forall \phi \in \Phi.$$

Plotting the footprints of the diminishing gaps for a given feature reveals the (possible) implication of its associated rule from other rules. For instance, Figure 3 shows two sets of footprints for $\phi^{(6)}$ and $\phi^{(11)}$. By starring the iteration when the rule of interest is actually learned, we can see that $r^{(6)}$ cannot be implied from the previous rules, since learning this rule dramatically closes the gap; on the contrary, $r^{(11)}$ can be implied from the starting seven or eight rules.

6.2.2 Analyzing Bigram Rules Given a rule-learning trace in the bigram setting, the analysis on efficiency and memorability, as well as feature entanglement, remains the same. However, every trace from the bigram model is generated as a continuation of the unigram learning: the bigram student is initialized from the latest unigram student (Section 5.2). This implies that the bigram rule set is initialized from the unigram rule set, rather than an empty set. MUS-ROVER uses the extracted bigram rules to overwrite their unigram counterparts — rules with the same features — highlighting the differences between the two language models. The comparison between a bigram rule and its unigram counterpart is key in recovering rules that are otherwise unnoticeable from the bigram rule alone, such as "Parallel P8s are avoided!" Therefore, MUS-ROVER emphasizes the necessity of tracking a series of evolving n-grams, as opposed to learning from the highest possible order only.

7 Discussion and Future Work

This paper presents MUS-ROVER that outputs musical knowledge instead of requiring it as input. It behaves like a music theorist, rather than a composer. Historically, a great theorist does not necessarily imply a great composer, and a great composer may not require music theory training to produce a masterpiece. It is not clear whether *any* set of rules, no matter how complex, can guarantee the composition of a great piece. Therefore, we do *not* judge MUS-ROVER by its compositional output; our aim is to demonstrate the extent to which rules can capture the decisions of great composers.

MUS-ROVER takes a first step in automatic knowledge discovery in music, and opens many directions for future work. Its outputs — the rule book and the learning traces — serve as static and dynamic signatures of an input style. We plan to extend MUS-ROVER beyond chorales, so we can analyze similarities and differences of various genres through these signatures, opening opportunities for style mixing. Moreover, while this paper depicts MUS-ROVER as a fully-automated system, we could have a human student become the generative component, interacting with "the teacher" to get iterative feedback on his/her compositions.

Acknowledgements

We thank Professor Heinrich Taube, President of Illiac Software, Inc., for providing Harmonia's MusicXML corpus of Bach's chorales.[6]

References

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(8):1798–1828.

Bengio, Y. 2009. Learning deep architectures for AI. *Found. Trends Mach. Learn.* 2(1):1–127.

Biles, J. 1994. Genjam: A genetic algorithm for generating jazz solos. In *Proc. ICMC*, 131–131.

Cope, D. 1996. *Experiments in musical intelligence*, volume 12. AR editions Madison, WI.

Dash, S.; Malioutov, D. M.; and Varshney, K. R. 2015. Learning interpretable classification rules using sequential rowsampling. In *Proc. ICASSP*, 3337–3341.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Proc. NIPS*, 2672–2680.

Gutmann, M., and Hyvärinen, A. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. AISTATS*, 297–304.

Malioutov, D., and Varshney, K. 2013. Exact rule learning via boolean compressed sensing. In *Proc. ICML*, 765–773.

Merz, E. X. 2014. Implications of ad hoc artificial intelligence in music. In *Proc. AIIDE*.

Mozer, M. C. 1994. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Conn. Sci.* 6(2-3):247–280.

Pape, A. D.; Kurtz, K. J.; and Sayama, H. 2015. Complexity measures and concept learning. *J. Math. Psychol.* 64:66–75.

Rajanna, A. R.; Aryafar, K.; Shokoufandeh, A.; and Ptucha, R. 2015. Deep neural networks: A case study for music genre classification. In *Proc. ICMLA*, 655–660.

Rohrmeier, M., and Cross, I. 2008. Statistical properties of tonal harmony in bachs chorales. In *Proc. ICMPC*, 619–627.

Simon, I.; Morris, D.; and Basu, S. 2008. MySong: automatic accompaniment generation for vocal melodies. In *Proc. CHI*, 725–734.

Taube, H. 1999. Automatic tonal analysis: Toward the implementation of a music theory workbench. *Comput. Music J.* 23(4):18–32.

Varshney, L. R. 2013. To surprise and inform. In *Proc. ISIT*, 3145–3149.

Vilhena, D. A.; Foster, J. G.; Rosvall, M.; West, J. D.; Evans, J.; and Bergstrom, C. T. 2014. Finding cultural holes: how structure and culture diverge in networks of scholarly communication. *Sociol. Sci.* 1:221–238.

[6] `http://www.illiacsoftware.com/harmonia`