Applying Learning Algorithms to Music Generation

Ryan Lichtenwalter
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana 46556
rlichten@cse.nd.edu

Katerina Zorina
Dept. of Musicology
Catholic University of America
Washington, DC 20064
zorina@cua.edu

Nitesh V. Chawla
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana 46556
nchawla@cse.nd.edu

ABSTRACT

Several automated composition systems have arisen that generate or facilitate understanding of blues chord progressions, jazz improvisation, or classical pieces. Such systems often work by applying a set of rules explicitly provided to the system to determine what sequence of output values is appropriate. Others use pattern recognition and generation techniques such as Markov chains. We propose a system in which sliding window sequential learning techniques are used to generate rules that correspond to a training set of musical data. This approach has the dual advantages of greater generality than explicitly specifying rules to a system and the ability to apply effectively a wide variety of powerful existing learning algorithms. We present the design and implementation of the composition process. We also illustrate several reasonably successful samples of its output and discuss ways in which it can be improved with additional work.

Categories and Subject Descriptors

J.5 [**Computer Applications**]: Arts and Humanities—*music*

General Terms

Algorithms, Human Factors, Languages

Keywords

data mining, machine learning, classification, sliding window, sequential learning, music, autonomous composition

1. INTRODUCTION

This paper represents a cross-disciplinary approach to leverage techniques germane to the domain of computer science to accomplish significant feats in the domain of fully autonomous music composition. The challenge presented is in the same class of challenges as the Turing test, wherein a computer seeks to successfully emulate human thought processes or patterns. Successful knowledge discovery in a large sample of existing compositions such that software can learn how to write a musically appealing composition is interesting; not only may it unveil patterns unanticipated by human analysis, but successful application techniques within the field of music are extensible to other problems in the Turing test class. Carrying knowledge discovery to such a degree that an expert on Bach chorales cannot distinguish between an unfamiliar chorale actually composed by Bach and a chorale composed by software represents a major accomplishment, in the same sense that passing the standard Turing test requires much more than a correct understanding of semantics. Although the vocabulary for tonal music is minimal, the grammar rules for tonal music composition are incredibly complex; there are so many rules that it may be difficult to specify them all without enumerating all existing compositions, both valid and invalid by aesthetic measure.

The approach consists of three major phases: data preprocessing for converting to a feature vector representation suited to machine learning, data analysis including rule inference, and nondeterministic music composition. The main theoretical challenge faced by this approach lies in the second and third steps: to produce rules that are both sufficiently general and relevant to the nature of tonal music, and also to make the best use of these rules to produce musical works that qualitatively resemble tonal music as described by the training set.

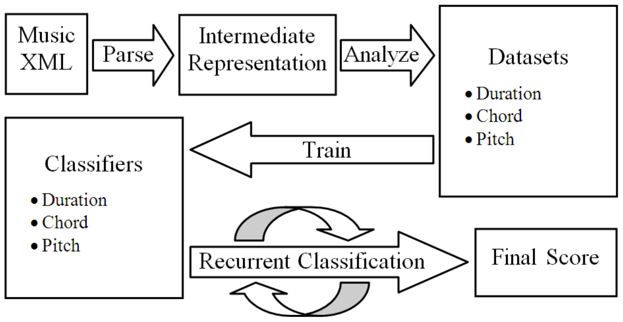

Figure 1: The Composition Process

Specifically, we apply the techniques of machine learning and data mining to the task of music composition with a large set of musical data from which standard algorithms can learn in a recurrent sliding window style. By applying different classifiers to various musical features to predict or generate the upcoming musical event, it is possible to produce different compositions. As a further element of curiosity, the compositions are audible embodiments of the inductive biases of the classifiers used to generate them.

Section 2 covers work related to autonomous composition. Section 3 provides information about the musical source data we chose to use, our method of transforming this source data to a form whereby traditional learning algorithms could be applied, and the construction of datasets. Section 4 describes the classification task. Section 5 delivers results and section 6 concludes. Section 7 provides ideas for future work based on many modifications and extensions to the final product highlighted here.

2. RELATED WORK

In [7], David Cope discusses his recent music composition tool, entitled Gradus. It is based on the principle that a melody can only be supported by certain harmonies, and operates over a very small subset of music. It reads a collection of 50 sample models in the form of short musical exercises and derives a small set of rules.[1] Using backtracking and recursion, Gradus is able to compose simple contrapuntal pieces; however, Cope admits that it does not initiate composition on its own, in that a user needs to explicitly supply the correct cantus firmus[2] variable. Although Gradus is capable of extracting cantus firmi from its own compositions, it remains to be seen how these differ from user-supplied cantus firmi. Cope's other works are primarily directed at defining tools to help composers better analyze music, or to enter musical scores more efficiently into a computer [6], [5].

The CHORAL project [12] uses traditional choral composition algorithms to harmonize existing chorales. It works in the following manner: given an existing bass line and soprano line, select the sounding chord and interpolate the inner voices. This procedure has been used by composers like Bach, who sometimes left the mundane task of writing the inner voices to his assistants. These techniques are, however, limited in application to four-part chorale style and therefore lack the generality that our approach has the potential to provide. The Impro-Visor tool [17] similarly requires a figured bass and improvises a melody atop it using probabilistic grammar modeling. Such efforts were primarily directed at monophonic jazz solos, and even within this limited context the output compositions do not seem very melodic to a trained musician.

Work by Smoliar in [24] is dated, but may elucidate the form a generalized framework for autonomous Schenkerian analysis could take. His concept of "musical event" is defined by "a single note, a sequence of events, or a simultaneity of events," and therefore, an entire piece could be represented as one musical event tree. A mathematically validated application within Smoliar's framework would represent a major accomplishment within the field of autonomous language processing. In his 1987 paper, Cope also suggests this as one possible approach to parsing music [4].

Friberg's paper aims for a human to define rules and a framework for automated composition [13]. Again, we stress that our application is fully autonomous and decides its own rules from a set of training examples. Once the challenge of fully automated composition is sufficiently addressed, Friberg's rules could be used for dynamic predictions based on sets of durations; furthermore, both Friberg's and Smoliar's ideas lend themselves to tree-based data structures for musical representation. Bresin and Friberg further addressed qualitative aspects of music in terms of post-processing on autonomously constructed compositions [1]. Wessel and Wright's interactive composition experimental procedure may also prove useful in the field of post-processing [26].

The commercial product Band-In-A-Box [16] allows a user to enter chords into a chart, and the product will improvise a melody (and other notes) at runtime and play the improvised music back. It also accepts a single audio file as a form of input and produces a list of chords. It does not dynamically compose its own chord lists. It is a tool for helping composers' imaginations during interactive composition sessions, but it is very limited in the sense of fully automated composition.

Several researchers have applied rule learning approaches to music generation in the past. A review of the application of machine learning methods for musical modeling and generation appears in [11]. Specifically, there have been other applications of sequential learning algorithms. Laine and Kuuskankare applied genetic algorithms to generate short musical sequences in [18]. Others have used Markov chains to recognize musical patterns and then apply those patterns to automatically generate music [25]. These and other approaches apply sequential learning techniques to the composition task, but none of which we are aware achieve the generality or flexibility of the recurrent sliding window approach provided here.

3. DATASET CONSTRUCTION

Autonomous music composition relies upon the ability to accurately represent and subsequently perform meaningful analysis on existing musical grammar. Unfortunately, these requirements are not as trivial as they may first seem. Researchers have conducted a large body of work in the area of music representation, access, and analysis. We encountered these auxiliary issues during our work and further advances in these capabilities will only improve the power and accuracy of all the methods that use them.

3.1 Source Data

We limited our analysis and predictions to Western tonal music, using exclusively Bach chorales as training data. The application of the approach thus excludes pieces that do not use an equally-tempered twelve-tone scale. One of the benefits of the approach, however, is that it can be trained on tonal music, modal music, and many other musical styles and genres with little or no alteration. The only requirement is that the input training data be of the same form as the goal output.

Arguably, tonal music, based upon the principles of well-defined durations and pitches, is the predominant form of music throughout the Western world. Within it, we focused on duration and pitch because these two concepts are the

[1]Example: If and only if chord B follows chord A in any training set, it is permissible for B to follow A in generated compositions.

[2]A preexisting melody used as a structural foundation for composition.

most ubiquitously understood, and the most fundamental, of all musical elements. Composing scores as sequences of durations and pitches is not fully artistic, but it is a large step with results that are easily understood and analyzed by either a listener untrained in music or an expert. If ever automated composition of duration and pitch elements is successfully achieved, the next step would be to add the other elements so critical in defining the artistic nuances of music.

There are several existing standards for music encoding, representation, and transmission. Undoubtedly the most popular is MIDI, a binary format designed to contain musical information in a compact form friendly to digital instruments. Although a wealth of MIDI files is available, MIDI fails to convey many useful musical characteristics that are not strictly necessary for playback of the piece with a digital instrument. Parsing MIDI files to reconstruct the full musical data available in a score is cumbersome, error-prone, and incomplete. We thus chose to use MusicXML as source data. The intention of the MusicXML format is to preserve musical information as it would appear on a page of printed music, with the complete information necessary to perform a piece or apply music theoretic analysis to it. Further, most modern music notation programs are capable of importing, handling, and exporting MusicXML making it a convenient format to visualize and play input and output data.

The availability of a large collection of Bach chorales online in various formats [14] made them a natural choice. Using Bach chorales to construct the dataset also presented other advantages. Although it might be an interesting study to see the results of combining training data representing several composers or compositional types, it is not helpful in evaluating aesthetically the output produced by the composition system. The exclusive use of Bach chorales confined the generality of the compositional task greatly. We further limited the selection of training data to four-part chorales without instrumental accompaniment. Each part has only a single voice and all fall within the domain of scores on which one may perform Schenkerian analysis. In total, we trained on 157 chorales comprising 41,414 pitch and duration dataset instances, and 9,856 chord dataset instances.

3.2 Intermediate Representation

Representing music in a way such that information can be efficiently and logically retrieved is a nontrivial task. An entire field of study has emerged in representing music effectively [27], [8]. Out of this have come several useful representations [22], [9], [21]. We have created our own simple object model. The model not only allows for a lossless representation of duration, chord, and pitch information, but also provides a contextualized structure in which that information can be successfully analyzed. Performing analysis on transfer formats such as MusicXML would be not only extremely difficult, but extremely inefficient. The object model thus better serves the needs of the various scanning and analysis components of dataset generation over the musical score. Generation of the object model requires a single scan of musical data from MusicXML. The final product much more easily enabled constructing training datasets, constructing testing datasets on which to predict, and finally generating new pieces.

3.3 Refactoring Tonalities

While we recognize that there may be fundamental differences in the chord progressions and common intervals between differing modes, we assume that within a given mode, the tonalities may all be reduced to a single representative. Therefore, based on the key information encoded in the MusicXML, we transpose all pieces designated in a major key to C major and all pieces designated in a minor key to A minor.[3] By doing the transposition, we obtain datasets of larger size presumably without any sacrifices. All relative movements in every major key correspond to particular relative movements in C major. In effect, the compositional system ignores key signature by placing everything in the same initial key signature and generating a sequence of pitches and chords that are correct in a relative pitch sense, rather than in an absolute pitch sense. One trivial artifact that emerged in some of the sample compositions is that the pitch ranges for the various parts do not correspond to the real pitch ranges in the Bach chorales. This could, of course, easily be handled by more carefully transposing training set pieces to ensure that they do not move outside of viable part ranges.

It is important to note that the division between the pieces designated major and the pieces designated minor is not strictly correct or meaningful. Especially in Bach, there may be many modulations to minor keys in pieces considered to be in major key. The opposite is also true. It may therefore make sense in future work to either remove the artificial separation into major and minor datasets, or to add instances to the datasets based on the overarching mode of the key as revealed by Schenkerian analysis instead of the mode of the key designated by the MusicXML for the score. Despite the modal confusion that occurs in the segregated major and minor datasets, there seems to be some significance to the division. Though the datasets are of similar size and composition, the minor datasets enjoy significantly better pitch predictability but fall prey to significantly poorer chord predictability.

3.4 Sliding Window Datasets

Because of their fundamental significance in defining how a composition sounds, the notions of duration and pitch are strictly necessary in defining any representation of a musical composition. For this reason, MusicXML files will always contain at least a definition of notes comprising pitch and duration, and rests comprising duration. We always limited the set of features to those that could be constructed with basic duration and pitch information, as well as higher level properties of the measures, parts, and scores to which that information belonged. This guaranteed that missing values could only result at the beginning of pieces when the sliding window was not yet primed.

Early attempts at dataset construction were only capable of supporting the composition of scores without considering vertical sonorities. There was no method of predicting a sequence of chords into which predicted notes should fit. The result was very much like late Medieval music in which rhythms and pitches were composed for particular parts,

[3]The specific choices of C major and A minor are functionally arbitrary but are somewhat more convenient to examine because these keys contain no sharps or flats.

but the interplay of the parts was considered unimportant. To rectify this limitation, we added a third dataset to create classifiers able to predict valid chord progressions. This dataset, which is the result of the synthesis of information available as pitches in the MusicXML, is an effort to consider the important relationships that tonal polyphonic music maintains across parts. The particular chord of which a pitch is a member also serves as a feature in the modified pitch dataset.

The final iteration of the system creates 22 distinct datasets. There are 8 pitch datasets: one for each part in major and minor key. There are 12 duration datasets: one for each part in $\frac{4}{4}$, $\frac{3}{4}$, and $\frac{3}{2}$ time. There are 2 chord datasets: one for major key and one for minor key. In general, the determination about what datasets must be constructed and used is automatic based on the training data and composition requests supplied to the system.

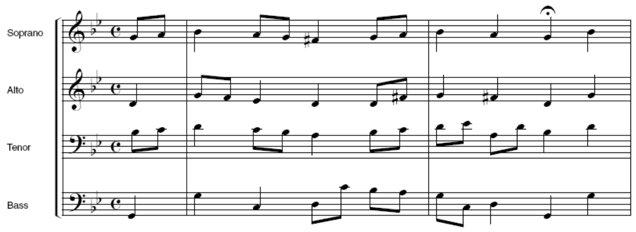

Figure 2: BWV 6.6 by J.S. Bach

3.4.1 Duration Datasets

In music, durations with close temporal locality exert great influences on each other. Sliding windows can capture this relationship by specifying that a duration n has as features all those durations preceding it within the window. Table 1 illustrates the features in the sliding window. The absolute temporal location of an event within a measure also has influences such that durations are affected not only by each other, but by the beat or tick at which they appear. The type of the measure, whether it is a pickup, first, last, pickup remainder, or middle measure, was included in the exploratory datasets and proved to have bearing in determining duration also. These features, which are not sequential in nature, are also included in the dataset.

The original approach that we took in building duration datasets was to divide the source data according to differing time signatures and then produce separate datasets for each time signature. The need for such data segmentation is undoubtedly obvious. Scores with $\frac{3}{2}$ time signature, for instance, could lead to rules predicting a whole note on beat 2. If such rules were applied at beat 2 in a $\frac{4}{4}$ measure, it would be disastrous. It may be prudent in future work to use time signature as a feature for a learning algorithm to make determinations as it wishes instead of creating separate datasets. In such cases, it is conceivable that rules would emerge contraindicating such illegal predictions. Work with a similar problem in the pitch dataset regarding part range confusion suggests it may be difficult to produce such contraindicating rules in practice. Further, there are other reasons to fully segment the datasets which correspond to stylistic differences in meter for which it might be very difficult to learn differentiating rules.

Because of major stylistic differences that may be evident across a selection of parts, it is also prudent to additionally segment duration datasets according to part. Although durations and duration sequences are relatively homogeneous across four-part chorales, this does not apply in orchestral scores with instruments of very different capabilities. Imagine, as a poignant example, what would happen when training a duration classifier on a piccolo part and using it to produce a tuba part.

3.4.2 Chord Datasets

Pitches in polyphonic music are applicable to individual notes, but are also influenced by and important in defining the interplay of multiple notes that are simultaneously audible. We can thus say that in addition to their own pitches, musical events have as a feature the chords of which they are members. Generating sequences of chords defines tonal polyphonic music, and thus it is necessary to create datasets and classifiers to handle music generation in the broad polyphonic context.

An ideal method, automatic Schenkerian analysis[4], remains an open problem in the field of computer music. Some of the difficulties with performing such analyses are explained well by Marsden [19]. Various methods to facilitate Schenkerian analysis and perform chord naming with results comparable to human analysis have been proposed [19], [24]. To our knowledge there have been no claims to achieve successful fully autonomous chord analysis using purely computational methods.

The inability to perform algorithmic Schenkerian analysis actually presented quite a problem for our work. Standard musical chord progressions dictate that only certain chords can follow particular sequences of previous chords. Since a chord goes a long way in dictating what collection of pitches is feasible for a given musical event, chord sequences seem like an excellent set of features with which to augment any dataset for pitch classification. The strength of the features in generating a good predictor is contingent upon the features being accurate. Furthermore, in data mining tasks there is the need for a large volume of source data to generate a good classifier or predictor. It is not feasible to manually analyze all of these chords so we were left with applying our best effort to what remains an open problem: autonomous chord analysis. Table 2 illustrates how the chord dataset would look using true Schenkerian analysis. In addition to these sequential features and the duration feature, the chord dataset also contains a feature describing the type of measure in which the chord resides as explained in section 3.4.1.

Rather than navigate through this tricky problem, we assumed that simple vertical sonorities can be as useful for pitch predictions as real chords. This assumption is not entirely correct, but it provides a good approximation with significantly less complexity. To encapsulate each unique vertical sonority while disregarding information irrelevant to Schenkerian analysis, we created a simple mapping from fifth degree indices to prime numbers and multiplied the primes to obtain a unique integer.

[4]Schenkerian analysis is a standard procedure used in analysis of tonal music first introduced by the nineteenth-century

Table 1: Duration Features for Bass Part (with Chord n)

Measure	Beat	Meas. Type	Tick	Dur. n-5	Dur. n-4	Dur. n-3	Dur. n-2	Dur. n-1	Duration n
0	4	pickup_first	576	-	-	-	-	-	eighth
0	4.5	pickup_first	672	-	-	-	-	eighth	eighth
1	1	middle	0	-	-	-	eighth	eighth	quarter
1	2	middle	192	-	-	eighth	eighth	quarter	eighth
1	2.5	middle	288	-	eighth	eighth	quarter	eighth	eighth
1	3	middle	384	eighth	eighth	quarter	eighth	eighth	quarter
1	4	middle	576	eighth	quarter	eighth	eighth	quarter	eighth
1	4.5	middle	672	quarter	eighth	eighth	quarter	eighth	eighth
2	1	middle	0	eighth	eighth	quarter	eighth	eighth	quarter
2	2	middle	192	eighth	quarter	eighth	eighth	quarter	quarter
2	3	middle	384	quarter	eighth	eighth	quarter	quarter	quarter

Table 2: Chord Features for Bass Part (with Chord n)

Measure	Beat	Measure Type	Duration	Chord n-5	Chord n-4	Chord n-3	Chord n-2	Chord n-1	Chord n
0	4	pickup_first	quarter	-	-	-	-	-	i
1	1	middle	quarter	-	-	-	-	i	i
1	2	middle	quarter	-	-	-	i	i	ii$^{\circ 6}$
1	3	middle	eighth	-	-	i	i	ii$^{\circ 6}$	V
1	3.5	middle	eighth	-	i	i	ii$^{\circ 6}$	V	V^{7}
1	4	middle	eighth	i	i	ii$^{\circ 6}$	V	V^{7}	i^{6}
1	4.5	middle	eighth	i	ii$^{\circ 6}$	V	V^{7}	i^{6}	vii$^{\circ 6}$
2	1	middle	eighth	ii$^{\circ 6}$	V	V^{7}	i^{6}	vii$^{\circ 6}$	i
2	1.5	middle	eighth	V	V^{7}	i^{6}	vii$^{\circ 6}$	i	iv^{7}
2	2	middle	quarter	V7	i^{6}	vii$^{\circ 6}$	i	iv^{7}	V
2	3	middle	half	i^{6}	vii$^{\circ 6}$	i	iv^{7}	V	i

Table 4: Translation of Pitches to Primes

Pitch	Eb	Bb	F	C	G	D	A
Fifth Degree	-3	-2	-1	0	1	2	3
Prime Index	5	3	1	0	2	4	6
Prime	13	7	3	2	5	11	17

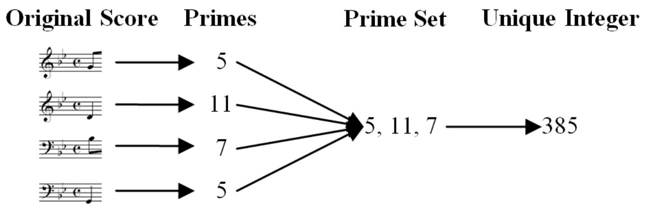

Figure 3: Vertical Sonority Translation Process

Disregarding octave information and duplicate pitches in each sequential vertical sonority and multiplying the primes that result from the translation process, a unique integer emerges to represent the sonority. Table 3 shows the appearance of the chord datasets actually used in training and generation. Some relationships between the integers and the Schenkerian analysis names in table 2 are apparent. Discrepancies in integer-to-name matches are the result of passing tones, which Schenkerian analysis disregards, but which are nevertheless present in the raw vertical sonorities. It is immediately clear that raw vertical sonorities create a much longer sequence of noisier features due to the passing tones.

German theorist Heinrich Schenker.

Although the raw vertical sonorities may present good features for pitch prediction, much more problematic is the effect of using vertical sonorities in predicting sequences of chords. A theoretically correct analysis of Bach yields a large number of cadences and other well-established sequences indicative of rules. Merely using vertical sonorities fails to disregard passing tones not strictly useful in determining the real chord sequence, thus fragmenting the set of supporting instances for the classifier. It also drastically increases the total number of possible class values in the chord features of the pitch and chord datasets. Section 5.1.2 suggests that there is a lot of room for improvement if the analysis of chords could be sufficiently improved. Indeed, with the high accuracy of the pitch predictor, improving the generation of chord sequences would be a major enhancement to the current implementation.

3.4.3 Pitch Datasets

In a sliding window context there are two primary sequential features for pitch. The first is the sequence of pitches in a particular part, which directly influence the upcoming pitch by dictating the interval necessary to reach the pitch. The second is the sequence of chords in a particular part, which will influence the upcoming pitch indirectly by first influencing the upcoming chord. When the order of the compositional effort is to first predict an upcoming chord and then to predict upcoming pitches within all the parts, the sequence of previous chords can be replaced by the chord within which the pitch to predict will fit. The current chord has a much more direct influence on pitch predictions, and experiments showed that the current chord feature has an information gain of 1.97 as compared to the much weaker

Table 3: Vertical Sonority Features for Bass Part (with Sonority n)

Measure	Beat	Measure Type	Duration	Chord n-5	Chord n-4	Chord n-3	Chord n-2	Chord n-1	Chord n
0	4	pickup_first	eighth	-	-	-	-	-	385
0	4.5	pickup_first	eighth	-	-	-	-	385	1870
1	1	middle	eighth	-	-	-	385	1870	385
1	1.5	middle	eighth	-	-	385	1870	385	1155
1	2	middle	eighth	-	385	1870	385	1155	442
1	2.5	middle	eighth	385	1870	385	1155	442	910
1	3	middle	eighth	1870	385	1155	442	910	7667
1	3.5	middle	eighth	385	1155	442	910	7667	15334
1	4	middle	eighth	1155	442	910	7667	15334	385
1	4.5	middle	eighth	442	910	7667	15334	385	1394
2	1	middle	eighth	910	7667	15334	385	1394	385
2	1.5	middle	eighth	7667	15334	385	1394	385	910
2	2	middle	quarter	15334	385	1394	385	910	7667
2	3	middle	half	385	1394	385	910	7667	385

information gain of 0.22 or less for each of the sequential w chord features in the sliding window.

The original approach that we took in building pitch datasets was to create only two separate datasets: one for major mode and one for minor mode. One problem with this approach is that part ranges overlap. When the classifier predicted soprano notes in the lower soprano range, it started inadvertently using training data from the alto range that would cause pitches to continue to descend lower into the alto range, into the tenor range, and eventually even into the bass range. The first attempt at fixing this was simply introducing a part feature to the pitch dataset. Information gain illustrated that the part feature was very weak, and practically it made very little difference. We finally settled on creating separate datasets for each part. Although this somewhat mitigates generality, it is a sensible solution. Even among music of the same general type, such as orchestra concertos, the piccolo will play entirely different lines in most cases from the tuba. To train the piccolo off both tuba lines and piccolo lines can only be counterproductive. Such an effort would be a clear instance of overgeneralization.

The complete set of features for the pitch datasets is visible in Table 5 along with a sample from the alto part of the excerpt in figure 2. The chord feature for a given pitch instance is always the chord at the time the pitch is sounded regardless of other chords that may occur while the pitch is sustained. This is a possible weakness in the current implementation.

Finally, although pitch and duration are linked [13], we thought it necessary to allow the duration of a pitch at time n to be a feature of the pitch dataset, since a combined (pitch, duration) class would be both computationally intensive and insufficiently general. That is, a count such as the number of times the raised fifth scale degree occurs on a triplet figure and similar counts are so specific that without an enormous dataset it would be difficult to derive meaningful rules using such a combined class. Further, duration is a feature of the pitch dataset, rather than the other way around, because the composition order predicts the duration class first during composition. Predicting the duration of musical events before the pitch is important because it is otherwise possible to select a pitch the reasonable durations of which do not fit within the rhythmic devices, measures, and phrases of the piece. The opposite is less likely. Given a relatively small set of possible durations and a much wider set of possible pitches, it is efficacious to determine the most restrictive information first.

4. MUSICAL CLASSIFICATION

4.1 Recurrent Sliding Windows

Music is partially related to and determined by broad static features. These might include the name of a part or instrument, the time signature, the tempo, and the type of measure. Many of these either serve as non-sequential features in our datasets either by segregating the data into distinct sets or by appearing a single time each instance to provide guidance to classifiers. Such information is related to the object-property traits of music. A score has parts, each of which has measures, each of which has additional divisions and descriptive information. Observing these objects can help in training classifiers, but the power of such features is limited. Most of the defining characteristics of music are dynamic: sequences of different durations, chords, and pitches.

In general, training classifiers on the non-sequential features can be handled by a large body of supervised non-sequential learning algorithms. Many of these have different inductive biases that can result in a broad range of performance characteristics on different datasets. Decision trees, artificial neural networks, genetic algorithms, support vector machines, and a host of others are available for non-sequential tasks. A large variety of ensemble methods and optimization techniques further increases the number of options. Our goal was to provide a generalized framework to provide this huge selection of classifiers with information encoded in the standard non-sequential format that would still carry the extreme power of the dynamic features of music. This information, which appears in the dataset instances as a window from event n-1 to event n-w, captures a given portion of a musical duration, chord, or pitch sequence at each time unit.

We sought to use these standard data mining classifiers to predict characteristics of a yet uncreated musical event n based on these characteristics. The size of the existing event vector w is a parameter defining the bounds of sliding win-

Table 5: Pitch Features for Alto Part (with Pitch n)

Measure	Beat	Meas. Type	Duration	Chord	Pitch n-5	Pitch n-4	Pitch n-3	Pitch n-2	Pitch n-1	Pitch n
0	4	pickup_first	quarter	385	-	-	-	-	-	D
1	1	middle	eighth	385	-	-	-	-	D	G
1	1.5	middle	eighth	1155	-	-	-	D	G	F
1	2	middle	quarter	442	-	-	D	G	F	$E^\flat$
1	3	middle	quarter	7667	-	D	G	F	$E^\flat$	D
1	4	middle	eighth	385	D	G	F	$E^\flat$	D	D
1	4.5	middle	eighth	1394	G	F	$E^\flat$	D	D	$F^\sharp$
2	1	middle	quarter	385	F	$E^\flat$	D	D	$F^\sharp$	G
2	2	middle	quarter	7667	$E^\flat$	D	D	$F^\sharp$	G	$F^\sharp$
2	3	middle	quarter	385	D	D	$F^\sharp$	G	$F^\sharp$	D
2	4	middle	quarter	385	D	$F^\sharp$	G	$F^\sharp$	D	G

dow W. The goal output, however, is not a single classification but the generation of a score, an entire series of classification or generative tasks. This recursive classification is one in which the results from the previous output serve as additional input in the updated window W for upcoming classification tasks. Because a blank score begins with no information whatsoever, it is necessary to seed it with the basic elements required by each of the sliding window classifiers.

After seeding a blank score with values sufficient to define an initial instance in the duration, chord, and pitch datasets for event n, it is possible to apply any non-sequential classifier. The output from this classifier is used to define a new instance to which the classifier may also be applied. In a score of length s, we will thus have predictions for each musical event from 2 to s: the duration of the homorhythmic event applying across all parts, the vertical sonority informing the acceptable event pitches in each of the parts, and the pitch of the event note in each part. Event 1 is currently seeded along with higher level information about the score such as the number and name of parts. The seeding could be replaced by the random generation of a set of first notes based on training data without loss of efficacy.

4.2 Classifier Application

Because a musical event may consist of a single note, multiple notes, or no notes at all, it may have associated with it the same set of characteristics that garnered individual datasets in section 3.4: duration, pitch, and chord. To produce a sensible musical event, all of these classes have to be generated appropriately and in a reasonable order. To this end, it is fortunately possible to consider many characteristics of musical events separately. In some cases, such as with duration and pitch, there is little dependence between characteristics, a fact that is substantiated in section 5.1.1 by the minimal decrease in entropy produced by the duration feature in predicting pitch. As an example, previous pitches and chords dictate new pitches and chords, but they do not dictate upcoming sequences of durations. Likewise, sequences of durations dictate appropriate upcoming sequences of durations, but do not as firmly dictate pitches.

Generating polyrhythmic scores that also fit into a generated sequence of chords is significantly more difficult than generating homorhythmic scores. The former task can proceed in one of two ways. The first is that a chord can be generated with a given duration and then individual parts can be constructed in the context of the chord. The second is that some subset of the parts can have pitches generated that help define the chord and then remaining parts can be filled. Both of these suffer from the difficulty that generating independent durations that fit into the encompassing chord duration is too restrictive and allowing durations to extend beyond the duration of the encompassing chord allows for inadvertently affecting upcoming chords. Essentially, it is difficult to create a score in which chords transition due to one part at a time changing pitch to cause the transition while still maintaining the integrity of the chords.

The specific order of the composition task was to first generate durations based exclusively on temporal information such as duration and position. Next, the duration is used as input to the chord predictor along with sequential chord features to predict a chord. The chord feature is then used to predict each of the parts of the score along with the same duration.

4.3 Learning Algorithms

Many of the characteristics of music are naturally suited to rule-based or decision-based algorithms. Consider the case of designing the rhythm to fit into a measure in common time. Suppose that the first three beats are quarter notes and the final beat is an eighth note. This accounts for 87.5 percent of the measure. In no cases will any of the training data ever support durations greater than 12.5 percent of the measure because such an instance would not complete a valid measure. It must also be true that no instances in training data will support a note value of less than 12.5 percent unless there is an accompanying instance in the training data that accounts for the remainder of the measure. If there were no such accompanying instance, the measure would be incomplete.

Placed in musical terms, if an eighth note is selected to complete the measure, 100 percent of the measure is accounted for. If a sixteenth note is selected, it means that there must be least one instance in the training data supporting such a prediction. Such an instance, however, will necessarily support the completion of the measure with either a series of note durations of lesser value or a final sixteenth note.

In support of these statements and the ability of the algorithms to learn, here are two of the specific rules that they produced. This is output from the Ripper algorithm, which often performed among the poorest of all the algorithms. Other rule-based and tree-based classifiers produced similar rules.

Rule 1:

```
if n_tick >= 672 then
  n_duration = eighth (74.0/0.0)
```

Rule 2:

```
if n-1_duration = dotted-quarter then
  n_duration = eighth (34.0/0.0)
```

This substantiates the earlier supposition that dotted quarter notes are frequently followed by eighth notes. The rule illustrates that in the Bach chorales, 100 percent of dotted quarter notes are followed by eight notes. There is no other option. The existence and subsequent automatic derivation of these types of rules is precisely what drives the accurate and valid duration component of the output compositions.

For pitches, rules are not as easily rendered because they tend to be softer. Unlike durations, where clearly illegal instances are easy to pick out, rules regarding pitch are less so. Still, especially given the backing of the n_chord feature, there is direct information about what set of pitches is likely to complete the specified vertical sonority. For a C major chord, either the $C^\sharp$ pitch or its enharmonic counterpart, $D^\flat$, are detrimental to the consonance of the chord to the point of being illegal. An even more obvious case is when the chord feature indicates a rest through all parts of the score. If there is such a rest, then each of the parts must also have a rest. These rules are illustrated below.

Rule 3:

```
if n_chord = 1 then
  rest (6.1)
```

The n_chord feature equals the multiplicative identity only when there are no pitches in the vertical sonority to donate their primes as factors.

Rule 4:

```
if n_chord = 1023 then
  4:B:0 (27.0/2.0)
```

In the cases when n_chord is 1023, the diminished triad composed of B, D, and F is the underlying chord. In this case, for the soprano part in the datasets based on minor chorales the pitch B4 is predicted with a great deal of confidence.

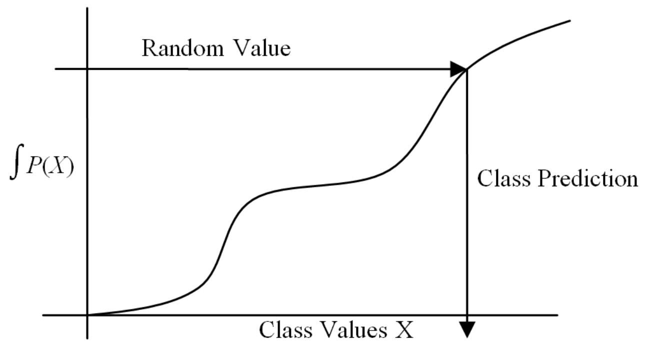

Figure 4: An illustration of the randomized prediction method.

4.4 Randomization and Non-determinism

In classical data mining tasks, the goal is to create a model that can most accurately classify and predict new instances. In a creative endeavor, this is neither the goal nor is it entirely desirable. Although it is important that a model effectively encompass the rules dictated by the musical source data, deterministically applying that model to produce stringently crafted results may severely limit the power of the compositional system.

Additionally, deterministic models in creative endeavors may be more subject to the problem of generating nonterminating repetitive sequences. Consider the extremely simple example of two rules DE→D and ED→E and an input of DE. Although a reasonably sophisticated model and the fact that there are many more features involved than just pitch will make such sequences highly improbable, they will not be impossible. In fact, the duration models used in this work generated precisely such sequences when applied deterministically.

The solution is simple: introduce non-determinism using randomization. The model still defines the rules that guide generation of the composition, but strong rules that can cause repetitive sequences are prevented from doing so. The approach for negating the effects of repetitive sequences while still observing the guidance of the training data involves basic statistics. First, we generated the probability distribution function for the predicted class value given the set of input features. Then, we constructed the related cumulative distribution function. Finally we generated a random number in a uniform distribution between 0 and 1 and intersected it with the cumulative distribution function. Figure 4 illustrates the concept:

This approach has the following important properties given an ideal model:

1. Class values that are most musically appropriate have the highest probability of occurring, guaranteeing that the next class is expected with respect to the input features.

2. Class values that are musically inappropriate have 0 probability of occurring. Essentially, the model contains a hard rule against such occurrences.

3. Class values that are uncommon will have a low probability of prediction, but will occasionally be predicted. These are soft rules.

The first two cases are somewhat banal and typical in data mining classifiers. The third is unique to creative endeavors. In the third case, such values add flavor and uniqueness to the music. If overused, the music loses cohesion, but such uncommon values are precisely what drive the creative output. Since randomization will allow such "poor" predictions to occasionally but infrequently occur, it is ideal.

In general, however, the model will not be ideal. When models with mediocre accuracy fail to cause overly repetitive sequences, using anything but the highest probability prediction from classifiers may be unnecessary and may instead decrease the quality of the output.

In practice, we initially observed many repetitive sequences without the application of randomization. In early implementations in which n_chord was not used as a feature, the prevalence of the tonic and dominant in the pieces caused those pitches to appear inordinately in output predictions. With the introduction of the n_chord feature in later implementations, pitch predictions no longer suffer from this problem. The variability of the sequences of vertical sonorities meant that little randomization was necessary. The extremely frequent appearance of quarter note durations in the chorales caused quarter note duration predictions to appear with almost 100 percent probability in prediction sequences. No level of classifier sophistication or application of boosting techniques solved this problem and it was necessary to use randomization for the duration predictor in the final implementation.

5. RESULTS

5.1 Performance Metrics

Before examining the actual compositional output that the system generated, we feel it both interesting and instructive to relate some basic learning metrics that we achieved based on our translation from music to features and classes. All of the results in this section describe properties of the datasets themselves and the performance of classifiers with classifications that have not been randomized. The necessity and results of randomization for duration predictions is explained in section 5.2. Because the recurrent sliding window approach is used, each classification represents both the class output of window W at position k and input for sequential feature n-1 at position k+1. Therefore, classification error is theoretically propagated. Unlike in many other sequential classification tasks, however, such propagation represents little damage. So long as each prediction is coherent in the context of the previous predictions, the error at any single point is relatively unimportant. For this reason, we currently have no results related to the way in which error is propagated throughout the predictions.

5.1.1 *Information Gain*

Table 6 illustrates the results of performing information gain analysis on the features we selected for determining pitch. The beat of the note to predict is by far the most important attribute. This corresponds to the fact that, especially toward the end of a measure, the location at which an event occurs is heavily constrained by the remaining time that may be legally allocated. The feature with the second highest information gain is the duration of the note before the note to predict. Dotted durations are often followed by a duration that fills the fractional portion of time. For instance, dotted quarter notes are followed by eighth notes and dotted half notes are followed by quarter notes. It is likely that the previous duration feature is a powerful predictor in other instances too, but these are less obvious. Finally, the remaining durations appear with an order of magnitude less information gain and their usefulness as predictors is questionable at best. We expected measure type to be an important feature because of its ability to discriminate the beat disparity in pickup measures and their counterparts at the end of pieces, but the data above show that this is not the case.

Table 6: Information Gain Analysis for the Duration Datasets

Feature	Information Gain
n Beat	0.51
n-1 Duration	0.25
n-2 Duration	0.06
n-3 Duration	0.04
n-4 Duration	0.03
n-5 Duration	0.03
Measure Type	0.05

Table 7 represents information gain across both the minor and major chord datasets. The information gain for all features in the minor datasets was slightly higher than for the major dataset features. This may suggest that pieces designated with minor keys follow a more rigid or predictable set of rules. The information gain for the previous chord is reasonable, and information gain for all of the chordal features is much higher than for the temporal features. This suggests that the appearance of a particular chord is generally independent of measure position and position in the piece. Of course, this is musically incorrect. Many rules exist that tie chords to their locations in the score in the Bach chorales. Just to name a few, the last measure must contain the tonic chord, the second to last measure most frequently contains the dominant chord, the cadential six-four chord may only occur on a strong beat unless it precedes a half cadence, the tonic typically occurs on strong beats. The fact that considerations of temporal locale have so little effect in reducing entropy may suggest a high degree of confusion in the appearance of particular chord types on strong or weak beats. Most certainly, the information gain is also mitigated by the usage of vertical sonority names rather than true chord analysis results.

The information gain analysis for the pitch training set in Table 8 is also telling. By far the most significant entropy-reducing feature is the previous pitch. In tonal musical, the set of intervals that are acceptable at a given instance is highly constrained within the context of the 88 pitches available on the piano. The specific musical source data, Bach chorales, probably boosted the value of the information gain. Bach chorales are further constrained to vocal ranges, which

Table 7: Information Gain Analysis for the Chord Datasets

Feature	Information Gain
n-1 Chord	1.97
n-2 Chord	1.36
n-3 Chord	1.12
n-4 Chord	1.03
n-5 Chord	0.97
n Duration	0.28
Measure Type	0.14

are significantly smaller than instrumental ranges. Further, the size of common melodic intervals in the training data was no more than an octave.

Table 8: Information Gain Analysis for the Pitch Datasets

Feature	Information Gain
n Chord	1.83
n-1 Pitch	1.36
n-2 Pitch	0.80
n-3 Pitch	0.59
n-4 Pitch	0.48
n-5 Pitch	0.42
n Duration	0.08
Measure Type	0.04

Every pitch behind the previous pitch has a successively lower information gain. While they appear to be somewhat better predictors for pitch than duration, pitches n-2 to n-w significantly less important than pitch n-1.

5.1.2 Accuracy

Figure 5 illustrates the accuracy achieved by several common classifiers implemented in WEKA [15]. Specifically, the figure represents the performance of naïve Bayes, C4.5 [23], Ripper [2], and a nearest neighbor classification algorithm [20] across each of the three classification tasks required by the automated composition system. We obtained all accuracies by performing 10 variably seeded runs of 10-fold cross validation.

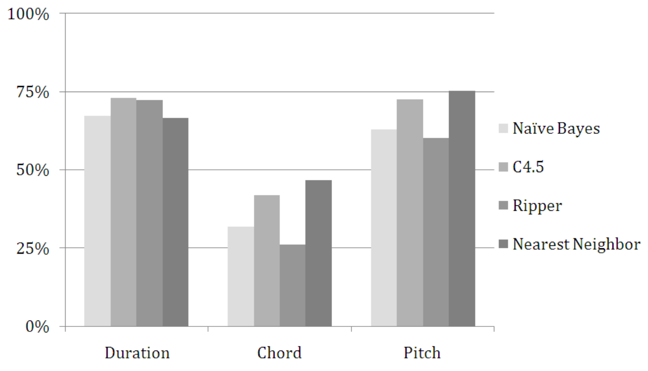

Figure 5: Accuracy of Standard Classifiers

It is immediately evident that the various musical classification tasks, and the datasets that support them, have different properties. The ease of the classification tasks varies greatly, and the inductive bias necessary to perform well on any given task is dissimilar.

The duration task, the first of the sources of input in the recurrent classifications, shows the highest accuracies across all classifiers. In fact, the values in figure 5 include both $\frac{3}{4}$ and $\frac{4}{4}$ datasets, but not the $\frac{3}{2}$ dataset due to data sparseness. In the $\frac{4}{4}$ datasets, classification was generally easier, with an optimal 10-fold cross validation accuracy of 75.8 percent by C4.5. The $\frac{3}{4}$ dataset only allowed for a 70.3 percent classification accuracy by C4.5. The performance disparity between $\frac{3}{4}$ and $\frac{4}{4}$ data was systematically comparable at approximately 6 percent for all algorithms. The optimal classification algorithm for durations was C4.5. Boosting, bagging, ensemble methods, parameter tweaking, and a variety of other algorithms did not improve this value. The $\frac{4}{4}$ soprano dataset yielded the highest accuracy to C4.5, where that algorithm achieved 80.7 percent.

The chord classifiers achieved much lower accuracies than their duration and pitch counterparts. There are several explanations for the low accuracy, many of which have already been partially provided. Because of the rudimentary nature of chord reckoning, the chord datasets are all very noisy. Consider that there are only a few dozen separate chord names under Schenkerian analysis that one could reasonably expect in the Bach chorales; yet, there are hundreds of distinct vertical sonorities. In the chord dataset containing the pieces designated as minor, the chord dataset with the larger range of possible chord values, there were 236 distinct vertical sonorities. This number disregards inversions and pitch doubling while the few dozen distinct chord names even include inversions. The much smaller range of class values visible to a classifier operating on real chords should automatically improve accuracy just because its chances are higher. More musically important, however, is the set of rules that real chord progressions follow, which the noisy vertical sonorities follow less often. We have little doubt that the installation of a good chord analysis model into the system could drastically improve chord classification performance, possibly doubling it.

It is also true that many of the vertical sonorities predicted by the classifier in testing may have been musically valid in the context of the sequential features provided. Further analysis is necessary to determine to what extent the incorrect classifications were actually invalid classifications. To such an end, it would be helpful to define a metric capable of indicating, beyond mere accuracy, the musical appropriateness of a particular model based on some of the output it provides. For the classification task, it would then be ideal to maximize this appropriateness metric in preference to accuracy.

The inductive bias apparently necessary to classify chords well differs starkly from that of durations and pitches. For chords, the Ripper rule algorithm performs abysmally, worse even than naïve Bayes. The C4.5 algorithm does well, but significantly better than any others is the nearest neighbor rule learning algorithm. The best accuracy obtained by this algorithm was 47 percent across both chord datasets. The chord dataset corresponding to pieces marked in minor key allowed for 44.6 percent accuracy while the major key chord

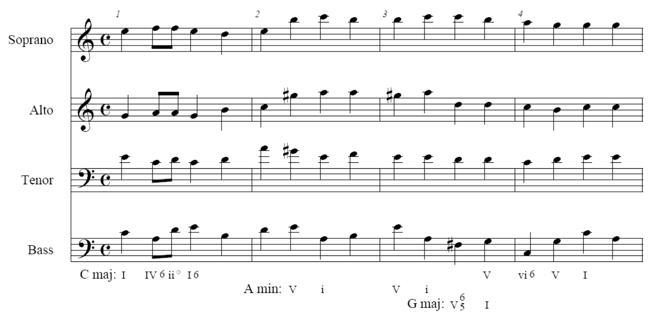

Figure 6: First Four Measure of Major Key Output

dataset admitted 48.9 percent accuracy. Various parameter tweaking and attempts with classifier ensembles failed to yield significantly better results. Although the discovery of the efficacy of the nearest neighbor algorithm allowed for over 20 percent improvement with respect to Ripper and 5 percent with respect to C4.5 it is likely that the only way to further improve classification accuracy is to provide datasets with less noise and more restricted class values through real chord handling.

Finally, the pitch dataset achieved accuracies as high as 75 percent over all major and minor datasets. As with the chord dataset, the best classifier proved to be the nearest neighbor rule algorithm, which reached 74.1 percent accuracy on the minor dataset and 76.3 percent accuracy on the major dataset for an average accuracy of 75.2 percent. Overall, classification of new instances with the minor dataset was somewhat more difficult with 1 to 2 percent lower performance for all classifiers.

Something of interest and also somewhat of a mystery is the difference in classifier performance across the part datasets. Classification accuracy proceeds from highest to lowest in the order bass part, soprano part, alto part, tenor part. All classifiers for both the major and minor datasets exhibited this characteristic with no exceptions. In some cases, the differences were extreme. The bass part classifications were 5 to 10 percent more accurate than the inner voice classifications. The soprano classifications were also significantly more accurate than the inner voice classifications. This is somewhat surprising because the bass part has the largest range of the four, while the inner voices have comparatively restricted ranges. This would seem at least superficially to make classification more difficult because a wider range of values exists for the class. The bass and soprano parts do enjoy some rules that do not hold for the inner voices. Soprano intervals are theoretically more predictable because they must be melodic. There are fairly stringent rules about skips and leaps. They should be few in number and they should be followed by step movement in the opposite direction. The bass must land on the root of the tonic chord in the last measure, although this rare special occurrence cannot have boosted accuracy so much. Still, the inner voices are expected to move relatively little, which could serve to restrict the effective range of reasonable class values for the classifiers. Because we are uncertain of the weight of each of these effects, the reason for this discrepancy in classifier accuracy eludes us.

The pitch dataset was the only example that yielded better performance to classifier ensembles. A voting classifier consisting of C4.5, Ripper, and the nearest neighbor rules classifier produced accuracies that were approximately 1 percent higher than their simple nearest neighbor counterparts, for an optimal classifier accuracy of 76.2 percent across all pitch datasets. Boosting on the classifier ensemble improved this slightly, but required a longer time for classifier training.

5.2 Compositional Output

The compositions were generated by the classifiers using datasets with the largest number of instances. The time signature of all pieces present in this section is $\frac{4}{4}$ because there were only about 700 instances in each of the $\frac{3}{4}$ duration datasets. The $\frac{4}{4}$ duration datasets had an order of magnitude more instances and, probably as a result, a 6 percent better accuracy. Both minor key and major key pitch and chord datasets were used to generate one piece each. The major key chord and pitch classifiers achieved higher accuracies than the minor key classifiers. Overall, however, we judge the output of the minor key compositions to be more aesthetically appealing. Although early compositional output had problems related to duration predictions, we corrected these in the final iteration of the system and it now learns and produces rhythms that always fit precisely into containing measures.

In examining the excerpts from the two pieces given in figures 6, 7, and 8, it will be helpful to treat them as exercises in voice-leading, comparing them to the rules learned in introductory music theory courses from studying the very Bach chorales that served as the training set for this experiment. For these purposes, the criteria by which the experimental pieces are judged are: chord structure and functionality, dissonance and leading-tone resolution, voice doubling, voice range, intervallic progressions, cadences, and overall logic of phrasing.

The sample composition in figure 6 is an example of the output produced by using classifiers from the set of pieces indicated to be in major key. The first chord was provided to the system as a seed for the recurrent sliding window process. The chords in the first measure form an unusual progression, although it is noteworthy that the third chord functions as neighbor to the fourth chord. The second measure contains a delightful tonicization of the relative minor, A, proceeding as expected from the dominant to the tonic of the new key. The only problem is posed by the doubled G sharp, which results in improper resolution of the leading tone in the tenor voice, which moves down to E instead of up to A. After it is introduced, A minor lasts for three more beats in a characteristic progression - tonic, predominant, dominant, tonic - barring the unfortunate root position of the supertonic which exposes the tritone between B and F. On the bright side, this time the dominant chord is resolved quite by the book. After this occurrence of the A-minor chord, there is another tonicization, this time of the dominant of the original key. The V6/5/V, that is the first-inversion D-major seventh chord, is missing its third, but its resolution is impeccable: the leading tone goes up by half-step and the seventh of the chord, C in the soprano, resolves down by step. The next chord, which has the G in the bass, functions both as the tonic of G major and the dominant of C, to which it returns on the next beat through an incomplete vi6 chord. The next two beats bring about the authentic cadence, V-I in the original C major.

Not all the chords are functional within traditional Western harmony (see m. 2, b.1), and some suffer from excessive doubling while leaving out important parts of the chord (see m. 1, b. 4; m.2, b. 4; m. 3, b. 3; m. 4, b. 1). In these four measures and later in the piece the ranges of the voices tend to go too high, which is to be expected given that the original training set underwent transposition resulting in indiscriminate registral shifting. There is one instance of parallel fifths and one of parallel octaves between two adjacent voices (m.1, b. 2.5-3 between alto and tenor; m.3, b.4-m.4, b.1 between alto and tenor). There is one instance of parallel fifths between bass and alto (m.1 b. 1-2), and there are two instances of hidden parallel perfect intervals (m.2, b. 1-2 between soprano and bass; m.3, b.4-m.4, b.1). The melodic intervals in the soprano create a coherent and appealing line of music with frequent change of direction and only one leap, followed as expected by step and creating a consonant interval, sixth, between the outer notes (m. 2, b. 1-3). The texture is contrapuntal and features a lot of desirable contrary motion. Phrasing suffers from rhythmic monotony and a lack of strong cadence at the end of this four-bar segment.

Two of these problems, excessive pitch doubling and rhythmic monotony are easily explained. The pitch doubling results from the fact that each of the parts has its pitch predicted independently of the others. Therefore, even if the chord progression is correct and the vertical sonority that all of the pitch predictors receive is viable, they may independently choose to all select one pitch in the sonority. The secondary factor driving the pitches predicted for the parts is the sequence of previous pitches, which may often make it unlikely for all four parts to select the same pitch based on the chord information. Still, there is nothing more than fortune to prevent the parts from all containing a single pitch. This condition is easily rectified by modifying the information that the pitch predictors receive about the chord with the pitch predictions of other parts.

The rhythmic monotony is the result of the excessively strong showing of quarter notes in the chorales. The classifiers would achieve high accuracy simply by always predicting quarter notes, and the models somewhat reflect this. Even with the rhythmic predictions randomized according to the method of section 4.4, rhythms tend to be relatively static. Often, rhythms of interest appeared in the output scores after the first few measures as rarer sequential feature values, once they appeared, motivated more exotic predictions. Successes in improving the rhythmic models to overcome this quarter note prevalence would dramatically improve output.

The excerpt in figure 7 is in A minor, which is announced immediately by the seed root-position tonic chord on the first beat. The entire first measure and the first beat of measure two form an expansion of the tonic chord with incomplete dominants functioning as passing chords. The true dominant chord follows with a raised third in the bass, which

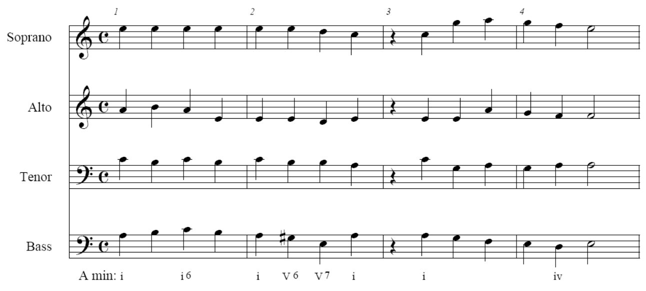

Figure 7: First Four Measures of Minor Key Output

resolves with a delay of one beat to the tonic on beat four of the second measure. The third measure opens with a rest, which marks a break from the introductory two measures of relative harmonic stasis. The next five beats feature a stepwise descent in the bass, which starts on the tonic and ends on the subdominant, filled in by passing chords. The fourth measure ends on a half-note chord, which is curiously composed of submediant constituents even though it does feature the fifth scale-degree in the bass, which is the seventh of this chord and also the bass note of a dominant chord that would be the next expected step in this progression.

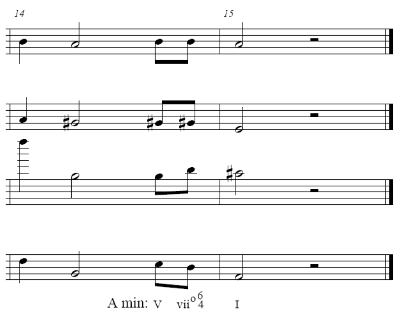

Figure 8: Final Two Measures of Minor Key Output

The last two measures of the piece in A minor are also worthy of particular attention. Measure 14 starts out with two non-functional harmonies. The fourth beat features the dominant chord followed by a diminished vii chord on the up-beat, which is best understood as a dominant seventh without its root. This is resolved beautifully to the tonic A chord with a Picardy third. This progression lacks proper resolution of both the leading tone and the seventh of the implied dominant chord. However, the final cadence reinforced by a major third, which does not occur on the tonic chord anywhere else in the piece, is a clear assimilation of Bach's style. This feature, along with the fact that the last chord is one of the few half-note chords in the piece and that the measure concludes with a rest, proves the specification of measure type to be very useful for both the duration and pitch classifiers. The classifier clearly recognized and learned from Bach's typical cadential progression in the last measure of chorales in minor keys.

Analysis of the two generated pieces allows us to draw several conclusions about the rules learned by the program. These conclusions are drawn not exclusively from the excerpts analyzed above but from the entire generated pieces.

The rules most likely learned by the classifiers are: the tonic chord should be almost always preceded by the dominant; the leading tone must be raised in secondary dominant chords; the leading tone must be resolved upward by step; excessive skips are to be avoided in all but the bass voice; duration sequences must fit precisely into the measures containing them. The classifiers may have learned that parallel perfect intervals, fifths and octaves, are to be avoided, but there is little theoretical support for this potential. They most likely did not learn the proper resolution and doubling rules as witnessed by unresolved sevenths of the chord and the rules of standard harmonic progression as there were many incomplete and non-functional chords. In the areas where the chord and pitch classifiers were most deficient, there is little doubt that properly reckoning chords would solve almost all problems.

5.3 Classifier Limitations

There are obvious limitations to the approach. We surmise that many of them can be overcome by carefully constructing a more complete set of the many features that may be extracted from music. One of the limitations yet to be discussed is the handling of patterns. Music is not always a flowing melody or harmony in which each note can travel to one of several possibilities. In motor rhythms or fluctuations between a restricted set of notes, there is precisely one possible "correct" value to predict. This is further complicated by the fact that such patterns usually do not continue forever. Representing music in such a way that patterns can be predicted by learning algorithms risks predicting in a manner that precludes deviation from the pattern. Departing from the pattern at an arbitrary time is also incorrect. We do not know whether the approach has even the potential to overcome such difficulties.

In essence, the problem is this: while learning algorithms applied in the manner of this paper can ferret out local rules in terms of a sequence of legitimate pitches or chords, it is much more difficult to train them to understand global or, at least, less local rules. The crux of the problem with our approach is having an understanding at both levels to overcome such locality issues. As we stated in our coverage of related work, we believe that it may be necessary to combine generation of music with rule discovery through learning algorithms and a direct application of specified rules and post-processing routines.

6. CONCLUSION

Researchers have dedicated a great deal of work to improving the efficacy of automated composition. Approaches have ranged from directly specifying rules to using sequential learning techniques such as Markov chains and results have served both casual research interests and commercial products. This work is different in its application of recurrent sliding windows to musical generation. This allows for the various characteristics of music, such as rhythm, pitch, chords, and others to each use a standard non-sequential learning algorithm with the optimal inductive bias.

Despite the power of the technique, there are also several difficulties. Perhaps the most severe and persistent of these is that, as it is implemented here, it is limited to creating unconnected sequences of chords and pitches. It has no knowledge of broader score concerns such as phrasing and repetition. The severely limited way in which chords are recognized and handled also greatly mitigates the accuracy of chord classifiers and the realism of the chords and chord progressions that are composed.

Nonetheless, the results are encouraging. After laying the tremendous groundwork necessary to implement a musical object model, create an extensible mapping from musical compositions to sliding window feature sets and classes, perform musical analysis, and interface digitally with the MusicXML representation, extending the system is easy. It is possible to generate music that looks and sounds reasonable based purely on rules discovered by the application of relatively suboptimal classifiers and rudimentary chord handling. Early successive improvements will undoubtedly dramatically improve output quality and realism.

In short, this paper serves as an introduction into a new way to approach the application of machine learning and data mining to the domains of musical understanding, recognition, and generation. By applying machine learning principles to the large sets of available musical data, it is possible to have a machine determine many more rules, and potentially more precise rules, than a human could ever specify. The recurrent sliding window approach provided here enables generalized handling of the compositional process and allows the application of a wide variety of classifiers for each of the broad, intersecting dimensions of music.

7. FUTURE WORK

are many ways in which the ideas we implemented could be extended to create what would probably be a much more powerful compositional model. The most obvious problem in the compositional output, which appears immediately without the slightest analysis, is that the score is not polyrhythmic. My work has focused on ways to develop the recurrent sliding window approach to make reasonable composition possible. This involved treating the many dimensions and interdependencies involved in music even when it is limited to the handling of durations and pitches. The best way to navigate the difficult interdependencies between chords, durations, and pitches of the parts that compose them would present a large breakthrough in this work. Another possible advance relates to the recurrent sliding window direction of movement. Forward sliding window approaches are not always the most efficacious. One example of this is illustrated by Dietterich in the context of a letter processing task [10]. In music, it is likely that the same applies. A compositional process through data mining is inductive in the sense that it applies a recurrent sliding window approach, but there must be a base case with which to start. This base case should have the highest stability possible with respect to the rest of the piece. It is often the case that the final notes and chords of a piece will have a strong relationship with what happened prior. The first notes and chords, however, may have little bearing at all on what should follow. They are often chords in pickup measures of the piece and do not relate in any telling way to the key signature. The final event in the piece is in this way more stable than the first event in the piece, making it a more logical starting point for the base case of the recursive inductive process.

Another potential improvement a recurrent sliding window approach starting at the end of the piece may garner relates to the way in which enharmonic spellings are handled in music. The spelling of a pitch is determined based on the note that comes after it rather than the note that comes before it. Thus, by starting at the end of the piece and working backward, it is easier to apply correct enharmonic spellings to note pitches.

Yet another possible improvement lies in constraining the musical representation of pitch. We originally opted for a completely accurate musical representation in terms of pitch. This meant supporting flats, sharps, double-flats, double-sharps, and potentially even greater pitch alterations for contemporary music. The presumption was that the precise enharmonic spelling of a note might be important in determining the pitch of notes nearby. We supposed that A-Bb-C$^\flat$ might predict differently from A-B$^\flat$-B in the sense that the first might typically go to pitch X and the second might typically go to pitch Y, where X and Y are not equal in pitch regardless of their predicted enharmonic spelling. The possibility of such fringe predictive power seems likely. At the same time, disregarding a rich musical representation and instead representing pitches as having only an octave and a chromatic degree, from 1 to 12, might produce better results. The size of the set of class values decreases dramatically. Such dramatic decreases may be even more significant when considering the effect the more constrained representation would have on the set of possible chords. If we say that the actual item of interest is the sound of a pitch rather than its appearance on paper, then it seems that such a constrained representation could be a much more powerful predictor, with a greater set of training instances confined to a smaller space of classes. It may also be true that enharmonic spellings actually do not add anything useful. If a particular enharmonic spelling can differentiate between correct and incorrect class predictions, and pitch sequences near the enharmonic spelling can differentiate the enharmonic spelling, then it may transitively hold that the pitch sequences carry all the predictive power that the unique spelling holds.

Along these same lines, an alternative handling of pitch might instead disregard absolute pitch altogether and use intervals [3]. Such an approach has the advantage of an even more generalized capture of the underlying musical information. Training on interval sequences rather than pitch sequences limits the total number of possible values that the pitch dataset sequential features and class can take to the relatively small set of intervals used in a particular set of music. Unfortunately, it can also complicate the simultaneous consideration of other important factors such as the range of the voice or instrument assigned to the part. Additionally, it is difficult to consider chords in the same way they are handled in this work when absolute pitch information is sacrificed to sequential intervallic features. Finally, viewing intervals such that absolute pitch information is completed disregarded may present other problems. In a piece in the key of C major, the sequence C-G corresponds to a fifth. It may be that the next interval should be a second, placing the next absolute pitch at A. For the sequence G-D, which is also a fifth, it may instead be that the next interval should be a fourth, placing the absolute pitch at G. Perhaps a better approach for extension to this work would be to supplement the existing sequential features with either a feature describing the size of the previous interval or even an additional set of sequential interval features. Learning algorithms could then determine if additional information about rising or declining intervallic motion is useful and optionally employ it.

Aside from the direct extensions of the system described here, there are numerous areas of related research where the approach could be helpful. One is the task of automated Schenkerian analysis, which continues to be problematic. By specifying sequential features such as previous chords or following chords in a sliding window, and by adding non-sequential features such as measure type, duration, and a host of others, it may be possible for a computer to learn many more rules than could ever be realized and provided to it. The essential problem is that Schenkerian analysis is largely a subjective endeavor, although there are formulaic aspects to it. Simply providing it with previously analyzed works could result in analyses that look substantially more like their human-created counterparts than is currently possible.

8. ACKNOWLEDGMENTS

Thanks to Margaret Greentree, who manages the jsb-chorales.net website. In addition to maintaining a wonderful repository of Bach chorales, she was willing to add MusicXML to her large list of offered formats. Using MusicXML directly, rather than extracting it from tags in the PDF files posted on the site, dramatically decreased noise in the data.

Dan Dugovic, as an undergraduate at the University of Notre Dame, assisted with the earliest attempts and with the review of related work.

9. REFERENCES

[1] R. Bresin and A. Friberg. Emotional coloring of computer-controlled music performances. *Computer Music Journal*, 24(4):44–63, December 2000.

[2] W. W. Cohen. Fast effective rule induction. In *Twelfth Interational Conference on Machine Learning*, pages 115–123, Tahoe City, California, USA, July 1995. Morgan Kaufmann.

[3] D. Conklin. Representation and discovery of vertical patterns in music. In *Second International Conference on Music and Artificial Intelligence*, pages 32–42. Springer-Verlag, 2002.

[4] D. Cope. An expert system for computer-assisted composition. *Computer Music Journal*, 11:30–46, 1987.

[5] D. Cope. The composer's underscoring environment: Cue. *Computer Music Journal*, 21(3):20–37, 1997.

[6] D. Cope. Computer analysis of musical allusions. *Computer Music Journal*, 27(1):11–28, 2003.

[7] D. Cope. A musical learning algorithm. *Computer Music Journal*, 28(3):12–27, 2004.

[8] R. B. Dannenberg. Music representation issues, techniques, and systems. *Computer Music Journal*, 17(3):20–30, 1993.

[9] R. B. Dannenberg. Computational music representation based on the generative theory of tonal music and the deductive object-oriented database.

Computer Music Journal, 27(3):73–89, 2003.

[10] T. G. Dietterich. Machine learning for sequential data: A review. In *Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30. Springer-Verlag, 2002.

[11] S. Dubnov, G. Assayag, O. Lartillot, and G. Bejerano. Using machine-learning methods for musical style modeling. *Computer Magazine*, 36(10):73–80, October 2003.

[12] K. Ebcionglu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.

[13] A. Friberg. Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15:56–71, 1991.

[14] M. Greentree. Jsbchorales.net: Bach chorales, April 2008.

[15] G. Holmes, A. Donkin, and I. H. Witten. Weka: A machine learning workbench. In *Second Australia and New Zealand Conference on Intelligent Information Systems*, pages 357–361, Brisbane, Australia, 1994.

[16] P. M. Incorporated. Band-in-a-box 2008 for windows, 2008.

[17] R. Keller and D. Morrison. A grammatical approach to automatic improvisation. In *Fourth Sound and Music Conference*, Lefkada, Greece, 2007.

[18] P. Laine and M. Kuuskankare. Genetic algorithms in musical style oriented generation. In *First IEEE Conference on Evolutionary Computation*, volume 2, pages 858–862, 1994.

[19] A. Marsden. Automatic derivation of musical structure: A tool for research on schenkerian analysis. In *The 8th International Conference on Music Information Retrieval*, pages 19–34, Vienna, Austria, 2007. Austrian Computer Society.

[20] B. Martin. Instance-based learning : Nearest neighbor with generalization. Master's thesis, University of Waikato, Hamilton, New Zealand, 1995.

[21] F. Pachet. An object-oriented representation of pitches, classes, intervals, scales and chords. *Premières Journées d'Informatique Musicale*, pages 19–34, 1994.

[22] S. T. Pope. Object-oriented music representation. *Organised Sound*, 1(1):56–68, 1996.

[23] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, USA, 1993.

[24] S. W. Smoliar. A computer aid for schenkerian analysis. *Computer Music Journal*, 4(2):41–59, Summer 9999.

[25] K. Verbeurgt, M. Dinolfo, and M. Fayer. Extracting patterns in music for composition via markov chains. *Innovations in Applied Artificial Intelligence*, pages 1123–1132, 2004.

[26] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. In *Conference on New Interfaces for Musical Expression*, volume 24, pages 44–63, Seattle, Washington, December 2001. National University of Singapore.

[27] G. Wiggins, E. Miranda, A. Smaill, and M. Harris. A framework for the evaluation of music representation systems. *Computer Music Journal*, 17(3):31–42, 1993.