Applying Learning Algorithms to Music Generation

Ryan N. Lichtenwalter[1], Katerina Lichtenwalter[1], and Nitesh V. Chawla[1]

The University of Notre Dame, Notre Dame, IN 46556, USA

Abstract. There exist several music composition systems that generate 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 models. These systems often suffer from mediocre performance and limited generality. We propose a system that goes from raw musical data to feature vector representation to classification models. We employ sliding window sequential machine learning techniques to generate classifiers that correspond to a training set of musical data. Our approach has the advantage of greater generality than explicitly specifying rules to a system and the potential to apply a wide variety of powerful existing non-sequential learning algorithms. We present the design and implementation of the composition system. We demonstrate the efficacy of the method, show and analyze successful samples of its output, and discuss ways in which it might be improved.

Key words: sequential learning, music, feature extraction

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. Successful knowledge discovery in a large sample of existing compositions such that software can learn how to write a musically appealing composition is interesting: it may unveil patterns unanticipated by human analysis and the techniques may be applicable to other creative endeavors. The problem is that music is a high-dimensional space with varying degrees of interdependency: durations, pitches, vertical sonorities, dynamics, articulations, etc. Further, its structure does not immediately suggest a single most efficacious approach to modeling. We demonstrate how sliding windows can be used to generate a powerful model, which can then be applied in a recurrent sliding window to compose musically interesting and relevant compositions.

The general approach consists of three phases: data preprocessing for converting to a feature vector representation suited to machine learning, data analysis

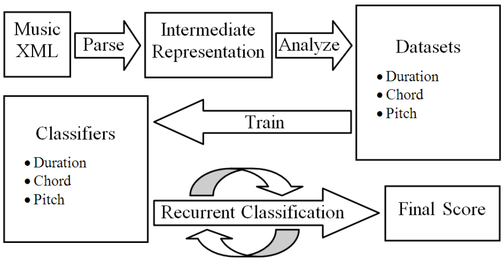

Fig. 1. The Composition Process

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.

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 and predict in a recurrent sliding window style. By applying different classifiers to various musical features to predict or generate upcoming musical events, 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.

2 Dataset Construction

We present the musical source data, some properties of that data and our method of transforming it to a “learnable form” by several standard classifiers.

2.1 Source Data

We limited our analysis and predictions to Western tonal music, using exclusively Bach chorales as training data. Tonal music is based upon the principles of well-defined durations and pitches. The availability of a large collection of Bach chorales online in various formats made them a natural choice. 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 is monophonic. Although MIDI is an established standard in digital music transmission, it fails to convey many useful musical characteristics and parsing it 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. In total, we trained on 157 chorales comprising 41,414 pitch and duration dataset instances, and 9,856 chord dataset instances.

Performing analysis on transfer formats such as MusicXML is not only difficult, but inefficient. We created our own simple object model, which better serves the needs of the various scanning and analysis components of dataset generation over the musical score. 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.

2.2 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. Using MusicXML key designations, we transpose all major key pieces to C major and all minor key pieces to A minor. By doing the transposition, we obtain datasets of larger size presumably without any sacrifices. One trivial artifact that emerged in some of the compositions is that the pitch ranges for the various parts do not correspond to the real pitch ranges in the Bach chorales. This is easily 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 that render these designations mostly useless. It may therefore make sense in future work to either remove the artificial separation into major and minor datasets, or to divide instances among datasets based on Schenkerian analysis key reckonings. 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.

2.3 Sliding Window Datasets

MusicXML files always contain at least a definition of notes comprising pitch and duration, and rests comprising duration. We 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. 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.

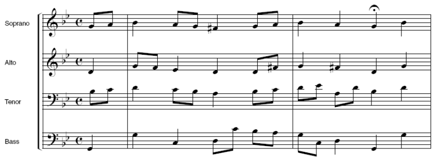

Fig. 2. BWV 6.6 by J.S. Bach

Table 1. Duration Features for Bass Part (with Duration n)

Position		Standard Features		Sliding Window Duration Features					
Meas.	Beat	Meas. Type	Tick	n-5	n-4	n-3	n-2	n-1	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

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 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 influences its duration. We also included the type of the measure (pickup, first, etc.,) in the datasets. These location features, which are not sequential in nature, are also included in the dataset.

We divided the source data according to differing time signatures and then produced separate datasets for each time signature. The need for such data segmentation is clear. 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 is possible that including time signature as a feature could produce rules contraindicating such illegal predictions. Our work suggests that this may not be entirely safe in practice. Further, there are other reasons to fully segment the datasets which correspond to subtle 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, we also segmented 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 what would happen when training a duration classifier on a piccolo part and using it to produce a tuba part.

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, remains an open problem in the field of computer music. Some of the difficulties with performing such analyses are explained well by Marsden [1]. Various methods to facilitate Schenkerian analysis and perform chord naming with results comparable to human analysis have been proposed [1], [2]. To our knowledge there have been no claims to achieve successful fully autonomous chord analysis using purely computational methods.

Because we choose not to attempt automated Schenkerian analysis and manual chord analysis is intractable, we instead rely on a surrogate for chord information as a powerful predictor of both upcoming chords and current pitches. This surrogate is the set of unique pitches disregarding registral information to compose a particular chord. Tables 3 and 4 demonstrate how the feature labels are constructed. The method is not musically correct, but it provides an approximation with significantly less complexity.

Table 2 illustrates how the chord dataset would look using true Schenkerian analysis. Table 3 shows the appearance of the chord datasets actually used in training and generation. Some relationships between the integers and the Schenkerian labels in Tab. 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 longer sequence of noisier features due to passing tones.

Although the raw vertical sonorities may present good features for pitch prediction, more problematic is the effect of using vertical sonorities in generating 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. Section 4.1 suggests that there is a lot of room for improvement if the analysis of chords could be sufficiently improved.

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. Since we first predict an upcoming chord and then predict upcoming pitches within all the parts, the sequence of previous chords can be replaced by the

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

Position		Standard Features		Sliding Window Chordal Features					
Meas.	Beat	Meas. Type	Dur.	n-5	n-4	n-3	n-2	n-1	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	V^7	i^6	vii$^{\circ 6}$	i	iv^7	V
2	3	middle	half	i^6	vii$^{\circ 6}$	i	iv^7	V	i

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

Position		Standard Features		Sliding Window Sonority Features					
Meas.	Beat	Meas. Type	Dur.	n-5	n-4	n-3	n-2	n-1	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

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

In a decision analogous to that of Sect. 2.3, we created separate major and minor datasets for each part. Different parts often play entirely different pitch sequences. Again imagine the piccolo and the tuba. This decision also prevents parts from leaving their ranges when a pitch overlaps the range of another part.

The complete set of features for the pitch datasets is visible in Tab. 5 along with a sample from the alto part of the excerpt in Fig. 2. The chord feature for a given pitch instance is always the chord at the time the pitch is sounded

Table 4. Translation of Pitches to Primes

Pitch	E$^\flat$	B$^\flat$	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

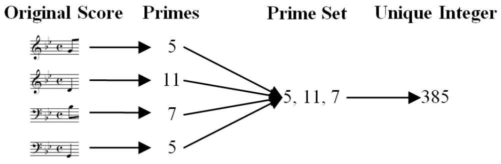

Fig. 3. Vertical Sonority Translation Process

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

Position		Standard Features			Sliding Window Pitch Features					
Meas.	Beat	Meas. Type	Dur.	Chord	n-5	n-4	n-3	n-2	n-1	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^b
1	3	middle	quarter	7667	-	D	G	F	E^b	D
1	4	middle	eighth	385	D	G	F	E^b	D	D
1	4.5	middle	eighth	1394	G	F	E^b	D	D	F$^\sharp$
2	1	middle	quarter	385	F	E^b	D	D	F$^\sharp$	G
2	2	middle	quarter	7667	E^b	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

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 [3], 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. Our decision to predict the duration of an event before its pitch is mostly arbitrary, but is also an effort to determine the most restrictive information first.

3 Musical Classification

3.1 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. Our goal was to create a generalized framework to provide classifiers with information encoded in the standard non-sequential format that would still carry the power of the dynamic sequential features of music. This information, which appears in the dataset instances as a window of events, captures a given portion of a musical duration, chord, or pitch sequence at each time unit.

We sought to use standard classifiers to predict characteristics of a yet uncreated musical event n based on these characteristics. 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 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 recurrent 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.

3.2 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 Sect. 2.3: 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 Sect. 4.1 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 order of the composition task as we defined it is 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.

3.3 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. Other rule-based and decision-based classifiers produced similar rules.

Rule 1: **if** n_tick >= 672 **then** n = eighth

Rule 2: **if** n-1 = dotted-quarter **then** n = eighth

This corroborates the supposition that dotted quarter notes are frequently followed by eighth notes. In fact, the rule statistics illustrate that in the Bach chorales, all 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

Rule 4: **if** n_chord = 1023 **then** 4:B:0

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

3.4 Randomization and Non-determinism

The power of the compositional system essentially lies in its flexibility, thus implying that deterministic application of any model can limit the expressiveness. While the model should effectively encompass the rules dictated by the musical source data, it should also carry sufficient expressiveness to avoid domination of deterministic 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 to 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: a) 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; b) Class values that are musically inappropriate have 0 probability of occurring. Essentially, the model contains a hard rule against such occurrences; c) Class values that are uncommon will have a low probability of prediction, but will occasionally be predicted. These are soft rules.

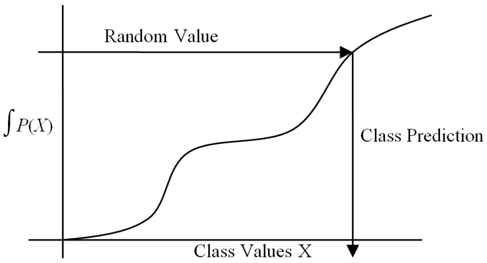

Fig. 4. An illustration of the randomized prediction method.

For chords and pitches, randomization proved to be unnecessary. For durations, the extremely frequent appearance of quarter note durations in the chorales caused quarter note duration predictions to appear with excessive probability in duration 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.

4 Results

We provide theoretical support for the efficacy of the approach and brief musical analysis of actual output samples.

4.1 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 Sect. 4.2. Because the recurrent sliding window approach is used, each classification represents both the class output of a window 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.

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	I.G.	Feature	I.G.
n Beat	0.51	n-4 Duration	0.03
n-1 Duration	0.25	n-5 Duration	0.03
n-2 Duration	0.06	Measure Type	0.05
n-3 Duration	0.04		

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 penultimate 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 Tab. 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,

Table 7. Information Gain Analysis for the Chord Datasets

Feature	I.G.	Feature	I.G.
n-1 Chord	1.97	n-5 Chord	0.97
n-2 Chord	1.36	n Duration	0.28
n-3 Chord	1.12	Measure Type	0.14
n-4 Chord	1.03		

which 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	I.G.	Feature	I.G.
n Chord	1.83	n-4 Pitch	0.48
n-1 Pitch	1.36	n-5 Pitch	0.42
n-2 Pitch	0.80	n Duration	0.08
n-3 Pitch	0.59	Meas. 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.

Accuracy. Figure 5 illustrates the accuracy achieved by several common classifiers implemented in WEKA [4]. Specifically, the figure represents the performance of naive Bayes, C4.5 [5], Ripper [6], and a nearest neighbor classification algorithm [7] 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.

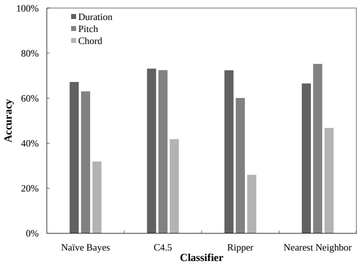

Fig. 5. Accuracy of Standard Classifiers

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 Fig. 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 nave 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 dataset admitted 48.9 percent accuracy. Various parameter tweaking and attempts with classifier ensembles failed to yield significantly better results.

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.

An interesting and curious result 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 the classifier ensemble improved this slightly, but required a longer time for classifier training.

4.2 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

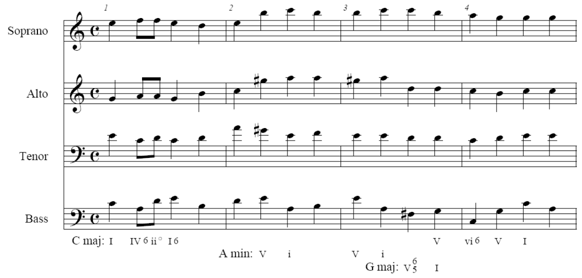

Fig. 6. First Four Measures of Major Key Output

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. Figures 6 and 7 are samples of output. We judge the work by 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 Fig. 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 is provided to the system as a seed for the recurrent sliding window process. Chord construction and chord sequences exhibit a wide range of acceptability with several progressions of chords being very nearly, but not quite correct due to excessive doubling or missing chord components. In these four measures and later in the piece the ranges of some voices tend to go too high, an artifact of indiscriminate registral shifting during training set transpositions. 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. 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 all independently choose to select the same pitch in the sonority, leaving out other necessary pitches. The sequential pitch features make it unlikely for all four parts to select the same pitch based on the chord information, but there are no rules 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, a form of skew in the training set. Often, rhythms of interest appeared after the first few measures of output as rarer sequential

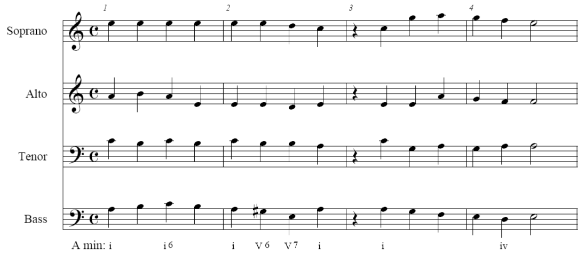

Fig. 7. First Four Measures of Minor Key Output

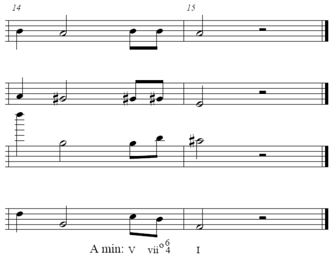

Fig. 8. Final Two Measures of Minor Key Output

feature values, once they appeared, motivated more exotic predictions. Standard differential sampling methods of handling skew caused a significant drop in performance, but more investigation is needed.

The excerpt in figures 7 and 8 are in A minor. Figure 7 shows desirable stepwise motion in reasonable sonorities. In Fig. 8, there is a good resolution to the tonic A chord with a Picardy third. The 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, suggests the specification of measure type is useful for both the duration and pitch classifiers. The classifiers 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 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. 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; there were many incomplete and non-functional chords. In the areas where the chord and pitch classifiers were most deficient, using proper chord names would solve almost all problems.

5 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. The compositional output compares very favorably to recent work in the field. 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. The 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.

6 Future Work

Our work presents several opportunities for enhancements and breakthroughs. Most obviously, the compositional system cannot currently create polyrhythmic scores; the task remains to find the optimal method to handle difficult musical interdependencies between chords, durations, and pitches of the parts that compose them. Another potential improvement lies in sliding window direction.

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 [8]. In music, it is likely that the same applies. The recurrent sliding window seed should have the highest stability possible with respect to the rest of the piece and this may be better achieved by starting on the final chord than the first chord. Working backward also allows for applying correct enharmonic spellings in stepwise pitch sequences. It may be efficacious to consider pitch differently in the work, opting for a less musically faithful consideration of pitches that collapses an infinite set of enharmonic spellings into a single true pitch. Along these same lines, an alternative handling of pitch might instead disregard absolute pitch altogether and use intervals [9]. Such an approach has the advantage of an even more generalized capture of the underlying musical information, but it can also complicate the simultaneous consideration of other important factors such as part ranges and chordal information. Aside from the direct extensions of the system described here, there are numerous areas of related research where the approach could be helpful. As an example, automated Schenkerian analysis may itself benefit from sliding window techniques.

7 Related Work

In [10], David Cope discusses his recent music composition tool, entitled Gradus. It reads data in the form of short musical exercises and derives a small set of rules. Using backtracking and recursion, Gradus is able to compose simple contrapuntal pieces, but does not initiate composition on its own in that a user must explicitly supply the correct cantus firmus variable. The CHORAL project [11] uses traditional chorale composition algorithms to harmonize existing chorales. It works by interpolating inner voices when given an existing bass and soprano line. Impro-Visor similarly requires a figured bass and improvises a melody atop it using probabilistic grammar modeling [12]. Work by Smoliar in [2] is dated, but may elucidate the form a generalized framework for autonomous Schenkerian analysis could take. Friberg's paper aims for a human to define rules and a framework for automated composition [3]. Once the challenge of fully automated composition is sufficiently addressed, Friberg's rules could be used for dynamic predictions based on sets of durations. Bresin and Friberg further addressed qualitative aspects of music in terms of post-processing on autonomously constructed compositions [13].

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 [14]. Specifically, there have been other applications of sequential learning algorithms. Laine and Kuuskankare applied genetic algorithms to generate short musical sequences in [15]. Others have used Markov chains to recognize musical patterns and then apply those patterns to automatically generate music [16]. 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.

8 Acknowledgments

Thanks to Margaret Greentree, who manages the jsbchorales.net website, for the MusicXML data.

References

1. Marsden, A.: Automatic derivation of musical structure: A tool for research on schenkerian analysis. In: The 8th International Conference on Music Information Retrieval, Vienna, Austria, Austrian Computer Society (2007) 19–34
2. Smoliar, S.W.: A computer aid for schenkerian analysis. Computer Music Journal **4**(2) (Summer 9999) 41–59
3. Friberg, A.: Generative rules for music performance: A formal description of a rule system. Computer Music Journal **15** (1991) 56–71
4. Holmes, G., Donkin, A., Witten, I.H.: Weka: A machine learning workbench. In: Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia (1994) 357–361
5. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, California, USA (1993)
6. Cohen, W.W.: Fast effective rule induction. In: Twelfth Interational Conference on Machine Learning, Tahoe City, California, USA, Morgan Kaufmann (July 1995) 115–123
7. Martin, B.: Instance-based learning : Nearest neighbor with generalization. Master's thesis, University of Waikato, Hamilton, New Zealand (1995)
8. Dietterich, T.G.: Machine learning for sequential data: A review. In: Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Springer-Verlag (2002) 15–30
9. Conklin, D.: Representation and discovery of vertical patterns in music. In: Second International Conference on Music and Artificial Intelligence, Springer-Verlag (2002) 32–42
10. Cope, D.: A musical learning algorithm. Computer Music Journal **28**(3) (2004) 12–27
11. Ebcionglu, K.: An expert system for harmonizing four-part chorales. Computer Music Journal **12**(3) (1988) 43–51
12. Keller, R., Morrison, D.: A grammatical approach to automatic improvisation. In: Fourth Sound and Music Conference, Lefkada, Greece (2007)
13. Bresin, R., Friberg, A.: Emotional coloring of computer-controlled music performances. Computer Music Journal **24**(4) (December 2000) 44–63
14. Dubnov, S., Assayag, G., Lartillot, O., Bejerano, G.: Using machine-learning methods for musical style modeling. Computer Magazine **36**(10) (October 2003) 73–80
15. Laine, P., Kuuskankare, M.: Genetic algorithms in musical style oriented generation. In: First IEEE Conference on Evolutionary Computation. Volume 2. (1994) 858–862
16. Verbeurgt, K., Dinolfo, M., Fayer, M.: Extracting patterns in music for composition via markov chains. Innovations in Applied Artificial Intelligence (2004) 1123–1132