A System for Computer Music Generation by Learning and Improvisation in a Particular Style

S. Dubnov[1], G. Assayag[3], G. Bejerano[2], O. Lartillot[3],

[1]*Ben Gurion University, Israel*

[2]*School of Computer Science, Hebrew University, Jerusalem, Israel*

[3]*Ircam (Institut de Recherche et Coordination Acoustique/Musique), Paris, France*

email: dubnov@bgumail.bgu.ac.il, jill@cs.huji.ac.il, lartillo@ircam.fr, assayag@ircam.fr,

Abstract

The paper deals with question of music modeling and generation, capturing the idiomatic style of a composer or a genre, in an attempt to provide the computer with "creative", human-like capabilities. The system described in the paper performs analyses of musical sequences and computes a model according to which new interpretations / improvisations close to the original's style can be generated. Important aspects of the musical structure are captured, including rhythm, melodic contour, and polyphonic relationships. Two statistical learning methods are implemented and their performance is compared in the paper: The first method is based on "Universal Prediction", a method derived from Information Theory and previously shown to be useful for statistical modeling in the musical domain. The second method is an application to music of Prediction Suffix Trees (PST), a learning technique initially developed for statistical modeling of complex sequences with applications in Linguistics and Computational Biology. Both methods provide a stochastic representation of musical style and are able to capture musical structures of variable lengths, from short ornamentations to complete musical phrases. Operations such as improvisation or computer aided composition can be realized using the system.

1. Introduction

The ability to construct a musical theory from examples is a great challenge, both intellectually, and as a practical means for a range of new and exciting applications. Machine learning is the process of deriving a model, such as a set of (stochastic) rules, from data samples. These models can be used for analysis and prediction, and, to a certain extent, they can generate acceptable original works that imitate the style of the masters, recreating certain aspects of the musical experience that were present in the original data set.

The process of composition is a highly structured mental process. Although it is very complex and hard to formalize, it is clearly far from being a random process. The task of this research is to try and capture some of the regularity apparent in the composition process by applying information theoretic tools to this problem.

Style Modeling

By Style Modeling, we imply building a computational representation of the musical surface that captures important stylistic features hidden in the way patterns of rhythm, melody, harmony and polyphonic relationships are interleaved and recombined in a redundant fashion. Such a model makes it possible to generate new instances of musical sequences that respect this explicit style (Assayag & al. 1999a). We adopt an analysis by synthesis approach, where the relatedness of the synthetic to the original may be evaluated to validate the analysis. We use unsupervised methods, requiring no human-intensive pre-processing, such as tagging or parsing. This facilitates the automatic learning process that may run directly on very large quantities of untagged musical data, abundantly available nowadays. Interesting applications include style characterization tools for the musicologist (Dubnov & al. 1998), generation of stylistic meta-data for intelligent retrieval in musical databases, convincing music generation for web and game applications, machine improvisation with human performers, and computer assisted composition.

One of the main purposes of learning in general is to create the capability to sensibly generalize. Composers, for example, are interested in finding out the creative possibilities of certain musical material, without necessarily "explaining" it. Statistical analysis of a corpus reveals some of the possible re-combinations that comply with certain constraints or redundancies typical of a particular style.

Mind-reading Machines

In the early 50's at Bell Labs David Hagelbarger built a simple 8 state machine, whose purpose was to play the "penny matching" game. This simple machine tried to match the future choices of a human player over a long sequence of random "head" or "tail" calls. Mind reading was done by looking at similar patterns in the opponent's past sequence that would help predict the next guess. The achieved rate of success was greater that 50% since human choices were not completely random and thus analysing patterns of previous choices could help foretell the future.

Inspired by Hagelbarger's success, Claude Shannon built a different machine with improved performance. An account of Shannon's philosophy on mind-reading machines can be found in (Shannon 1953).

It is important to note that had the model of the data sequence been known ahead of time, optimal prediction could have been achieved. The difficulty that needs to be tackled with almost all real-life situations is that the probability model for the data is unknown.

Music Generation and Style Replication

A generative theory of music can be constructed by explicitly coding music rules in some logic or formal grammar (Cope 1987, Ebicoglu 1986, Lidov & Gambura 1973). This approach is sometimes called an "expert system" or "knowledge engineering approach". A contrasting approach is the statistical learning or empirical induction approach. Several researchers have used probabilistic methods, notably Markov models, to model music (Pinkerton 1956, Brooks Jr. & al. 1993, Conklin & Witten 1995). Experiment by Brooks Jr. & al. (1993) showed that at very low orders (e.g., bigram), Markov model generated strings that do not recognizably resemble strings in the corpus, while at very high orders, strings from the corpus are simply replicated.

An interesting "compromise" between the two approaches is found in more recent works of Cope (1991). Cope uses a grammatical generation system combined with what he calls "signatures", melodic micro- gestures common to individual composers. By identifying and reusing such signatures, Cope was able to reproduce the style of past composers in reportedly impressive ways.

Predictive Theories in Music

Following the work of Meyer (1961) it is commonly admitted that musical perception is guided by expectations based on the recent past context. In music applications the model is unknown. Considering the amount of relevant past samples is difficult since the length of the musical context is highly variable, ranging from short figurations to longer motifs. Taking a large fixed context makes the parameters difficult to estimate, growing exponentially in the computational cost and data requirements. In order to cope with this problem one must design a predictor that can deal with arbitrary observation sequences and is competitive to a rather large class of predictors, such as variable length Markov Predictors. In this paper we use two approaches: We learn a universal predictor, and contrast its performance with that of the best model estimated from a restricted class of Markovian predictors.

2. Dictionary-based prediction

In our work we first present an exhaustive dictionary-based prediction method, which parses an existing musical text into a lexicon of phrases/patterns, called *motifs*, and provides an inference method for choosing the next musical object following a current past *context*. The parsing scheme must satisfy two conflicting constraints. On the one hand, one wants to maximally increase the dictionary to achieve better prediction, but on the other hand, enough evidence must be gathered before introducing a new phrase, so that a reliable estimate of the conditional probability is obtained. The secret of dictionary-based prediction (and compression) methods is that they cleverly sample the data so that most of the information is reliably represented by few selected phrases. In contrast, fixed order Markov models build potentially large probability tables for the next symbol at every possible context entry. The selective dictionary-based prediction method builds more complex variable memory Markov models to avoid this pitfall. Although at first it may seem that the two methods operate in a different manner, they both stem from similar statistical insights.

3. Exhaustive dictionary-based prediction

As we have explained above, we use dictionary-based methods to assess the probability of the next sample given its context. We turn to describe the first parsing and generation algorithm.

3.1. Incremental Parsing

We consider an incremental parsing (IP) algorithm that was originally suggested by Lempel and Ziv (1978) in the context of lossless compression. IP builds a dictionary of distinct motifs by one continuous left to right traversal of a sequence, sequentially adding to a dictionary *every* new phrase that differs by a single last character from the longest match that already exists in the dictionary. For instance, given a text {ababaa…}, IP parses it into {a,b,ab,aa,…} where motifs (dictionary entries) are separated by commas. The dictionary may be efficiently represented as a tree and the search and parse steps are equivalent to a process of growing that tree (see example below).

Probability Assignment

Assigning conditional probability $\hat{p}^{LZ}(x_{n+1} \mid x_1^n)$ to a symbol x_{n+1} given x_1^n as context is done according to the code lengths of the Lempel Ziv compression scheme. Let *c(n)* be the number of motifs in the parsing of an input *n*-sequence (or the number of nodes in the tree representation). Then, *log(c(n))* bits are needed to describe each prefix (a motif without its last character), and 1 bit to describe the last character (in case of a binary alphabet). For example, the code for the above sequence is (00,a),(00,b),(01,b),(01,a) where the first entry of each pair gives the index of the prefix and the second entry gives the next character. Ziv and Lempel have shown that the average code length *c(n)log(c(n))/n* converges to the entropy of the sequence with increasing *n*. This proves that this coding is asymptotically optimal. Since for optimal coding the code length is 1/probability, and since all code lengths are equal, we may say that, at least in the long run, all IP motifs have equal probability.

In tree representation, every node is associated with a string, being the characters on the arc leading from the root to that node. Each time that the parse reaches the node, it means that the node's string has occurred at the start of a phrase boundary. When this occurs, a child node is grown which corresponds to the appearance of a new phrase that is as continuation of the parent string, i.e. the offspring nodes are labeled by strings whose prefix is the label of the parent node.

Taking equal weights for all nodes in the tree representation $\hat{p}^{LZ}(x_{n+1} \mid x_1^n)$ will be deduced as a ratio between the cardinalities of the subtrees (number of subnodes) following the node x_1^n. As the number of subnodes is also the node's share of the probability space (because one codeword is allocated to each node), we see that the amount of code space allocated to a node is thus proportional to the number of times it occurred.

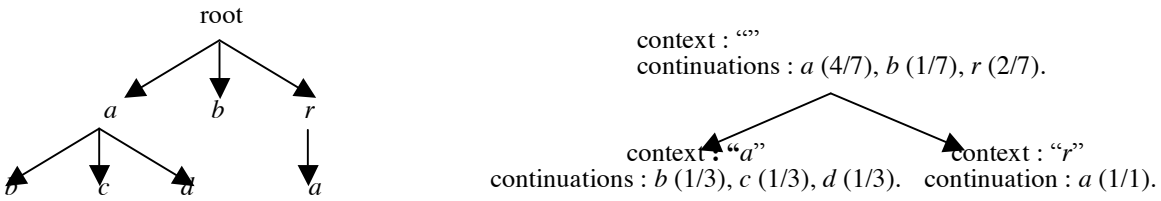
Thus, for a node labeled by string *c,* the conditional probabilities of the descendents are taken as the relative portion of counts of characters $N_c(x), x \in \{a,b\}$ appearing in all descendants (children nodes) of the current node c, setting $P_c(x) = \frac{N_c(x)}{N_c(a) + N_c(b)}$.

3.2 Example

Here is how IP is used to generate a sequence based on analysis of the very simple sequence "*abracadabra*". The generation process is incremental: The first letter is a continuation of the empty context "". The next letters are each a continuation of the longest context that is a suffix of what has been already generated. In our simple example, if the process first generates an *a* (with probability 4/7), then the next letter is a continuation associated with the context "*a*" (uniform over *b*, *c* and *d*) etc. For example:

P(generate "*abrac*") = P(*a*|"")P(*b*|***a***)P(*r*|*ab*)P(*a*|*abr*)P(*c*|*abra*) = 4/7 · 1/3 · 2/7 · 1 · 1/3.

The corresponding analysis tree and probability assignments are:



4. Selective dictionary-based prediction

Despite the fact that an IP predictor may asymptotically outperforms a Markov predictor of a finite order (Feder et al, 1992), the music sequences we are modeling in practice are of a finite length. Moreover, there is no motivation to parse the reference set on the fly, as IP does. Allowing one to use the whole sequence for estimation of the statistics may help improve the performance for realistically long sequences.

4.1 Prediction Suffix Trees

The Prediction Suffix Tree (PST) algorithm, named after the data structure used to represent the learned statistical model, was developed by Ron & al. (1996). A PST represents a dictionary of distinct motifs, much like the one generated by the IP algorithm. However, in contrast to the lossless coding scheme underlying the parsing there, the PST algorithm builds a restricted dictionary of *only* those motifs that both appear a significant number

of times throughout the *complete* source sequence, *and* are meaningful for predicting the immediate future. The framework underlying the approach is that of efficient lossy compression.

Selective Parsing

We describe an empirically motivated variant of the PST algorithm (Bejerano & Yona (2001)). It uses a breadth first search (BFS) for all motifs that comply simultaneously with the three requirements that follow and collects only these into our restricted dictionary:

- The motif is no longer than some maximal length *L*.
- Its empirical probability within the given data set exceeds a threshold *Pmin*.
- The conditional probability it induces over the next symbol differs by a multiplicative factor of at least *r* from that of the shorter contexts it subsumes, for at least one such next symbol (see below).

Probability Assignment

We define the empirical probability of a motif as the number of occurrences it has in our text divided by the number of times it could have appeared in it. For example the empirical probability of the motif "*aa*" within the sequence "*aabaaab*" is 3/6=0.5.

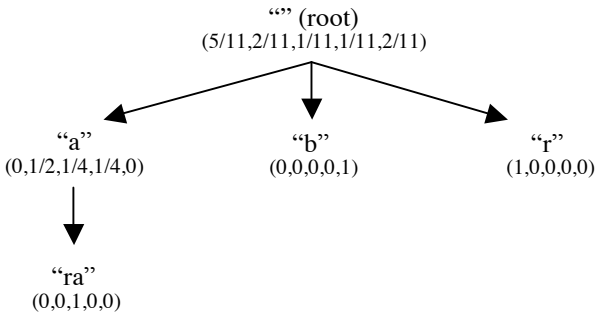
A conditional probability to observe a symbol after a given context is defined as the number of times this symbol comes immediately after that context, divided by the total occurrences of the context in our text. Thus, the conditional probability of seeing "*b*" after "*aa*" in the sequence "*aabaaab*" is 2/3.

And since the conditional probability of seeing "*b*" after "*a*" is 2/5, then the multiplicative difference between the conditional probability to observe "*b*" after "*aa*" and that after "a" is (2/3)/(2/5) = 5/3.

In the PST formulation the counts are incorporated in a slightly different manner. Using the notation of the previous section we set $\hat{P}_c(x) = (1 - |\Sigma| g)P_c(x) + g$, where $|\Sigma|$ is the size of the alphabet and g is the smoothing factor, $0 < g < 1/|\Sigma|$. Fractional probabilities are collected from all possible next symbol extensions in proportion to their actual counts, and then redistributed equally between all to achieve a minimal conditional probability for any continuation $\hat{P}_c(x) \geq g > 0$.

4.2 Example

Here is the PST analysis of "*abracadabra*", with $L = 2$, $Pmin = 0.15$, $r = 2$, and $g = 0$. With each node we associate the list of probabilities that the continuation may be, respectively, *a*, *b*, *c*, *d* or *r*:



Note, for example, that potential node "*ba*" failed the *Pmin* criterion, while potential node "*bra*" passed that but failed both the *L* and *r* criteria.

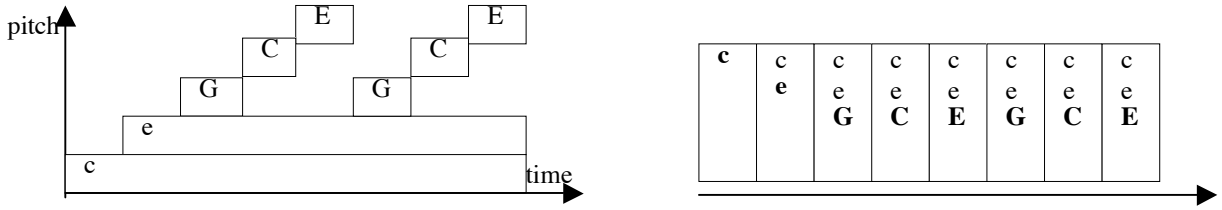
Generation proceeds much like in the IP resulting tree. However now, for example

P(generate "*abrac*") = P(*a*|"")P(*b*|***a***)P(*r*|*a****b***)P(*a*|*ab****r***)P(*c*|*ab****ra***) = 5/11 · 1/2 · 1· 1 · 1.

5. Music Representation

So far we have loosely used the term sequence do denote an ordered list of objects or symbols. In order to capture a significant amount of musical substance we shall, in a pre-analytic phase, cut the musical data (generally in Midi format) into slices, whose beginnings and ends are determined by the appearance of new events and termination of past ones. Every slice has a duration information, and contains a series of channels, each of which contains pitch and velocity information and possibly other musical parameters (these slices will be referred to as objects or symbols, and the set of possible symbols as the alphabet).

On the left side of the figure, for example, is the piano roll representation of the beginning of Bach's prelude in C, where lower case letters represent the third octave, and upper case letters the fourth one. The right side of the figure shows how this sequence is sliced into symbols. The bold letters represent beginnings of notes.



5.1 Enhancement Filters

Having defined the basic quantization method of symbols and the learning scheme, one may turn to music generation. However in light of some theoretical insights, as well as some preliminary tests, several improvements have been made to our system. These improvements consist of pre-analytic simplification, generative constraints, loop escape, and analysis-synthesis parameter reconstruction.

Pre-analytic Simplification. Real musical sequences - for example Midi files of a piece interpreted by a musician - feature fluctuations of note onsets, durations and velocities, inducing a complexity which fools the analysis: the alphabet size tends to grow in an intractable way, leading to unexpected failures, and poor generalization power. We have thus developed a toolkit containing five simplification filters:

- The *arpeggio filter* vertically aligns notes which are attacked nearly at the same time
- The *legato filter* removes overlap between successive notes
- The *staccato filter* ignores silence between successive notes
- The *release filter* vertically aligns note releases,
- And the *duration filter* statistically quantizes the durations in order to reduce the duration alphabet.

These features can be tweaked by the user, using thresholds (e.g. a threshold of 50ms separates a struck chord on the piano from an intended arpeggiated chord). Using the simplification toolkit, Midi files containing real performances can be treated. This particular kind of live musical data, full of idiosyncrasy, is of great value as a model for synthetic improvisation.

Generative Constraints. It is possible to specify constraints during the generation phase. At each cycle, if the constraint is not respected by the new symbol, the generation is canceled and a new symbol is tried. If no symbol is satisfying, the algorithm backtracks to the previous cycle, and more if necessary. For example, it is possible to specify a continuity constraint that imposes a minimal context size anytime during the generation phase, so as to avoid musical discontinuity in case of an empty context.

Loop Escape. If no prediction smoothing is applied, the generation phase may easily enter into an infinite loop. We introduce the concept of a context-generated subtree, which consists of the exhaustive set of all the

possible contexts that may be met after the present one. Practically, we just need the size of this subtree, which is computed only once before the generation phase. Its computation consists of marking, directly in the original tree, of all the possible future contexts and a counting of these marks. When the size of the context-generated subtree is below a user-specified threshold N, an N-order-loop is detected. The loop phenomenon is principally due to the fact that the generation phase searches for the maximal context, which is unique and proposes few alternative continuation symbols. In the case of a loop, we loosen this constraint and examine not only the maximal context, but also shorter ones, which escape the loop in most cases.

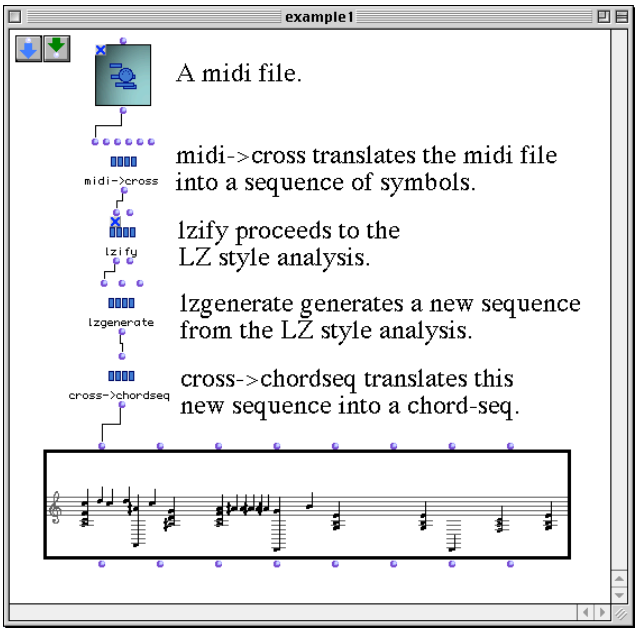
Analysis-Synthesis Parameter Reconstruction. In our formulation, a symbol is a Cartesian product of several musical parameters. In order to increase abstraction power in the analysis, we allow the system to discard some parameters - for example, the velocity. The retained parameters are called analysis information. However, in the generation phase, the discarded information cannot be retrieved. For example, if we choose to discard note durations and dynamics, we end up obtaining isorythmic and dynamically flat musical sequences that sound like the work of a musical box. Our solution is to store in the model the excluded information, called synthetic information. During the generation phase, synthetic information, such as expressivity, may be reconstructed.

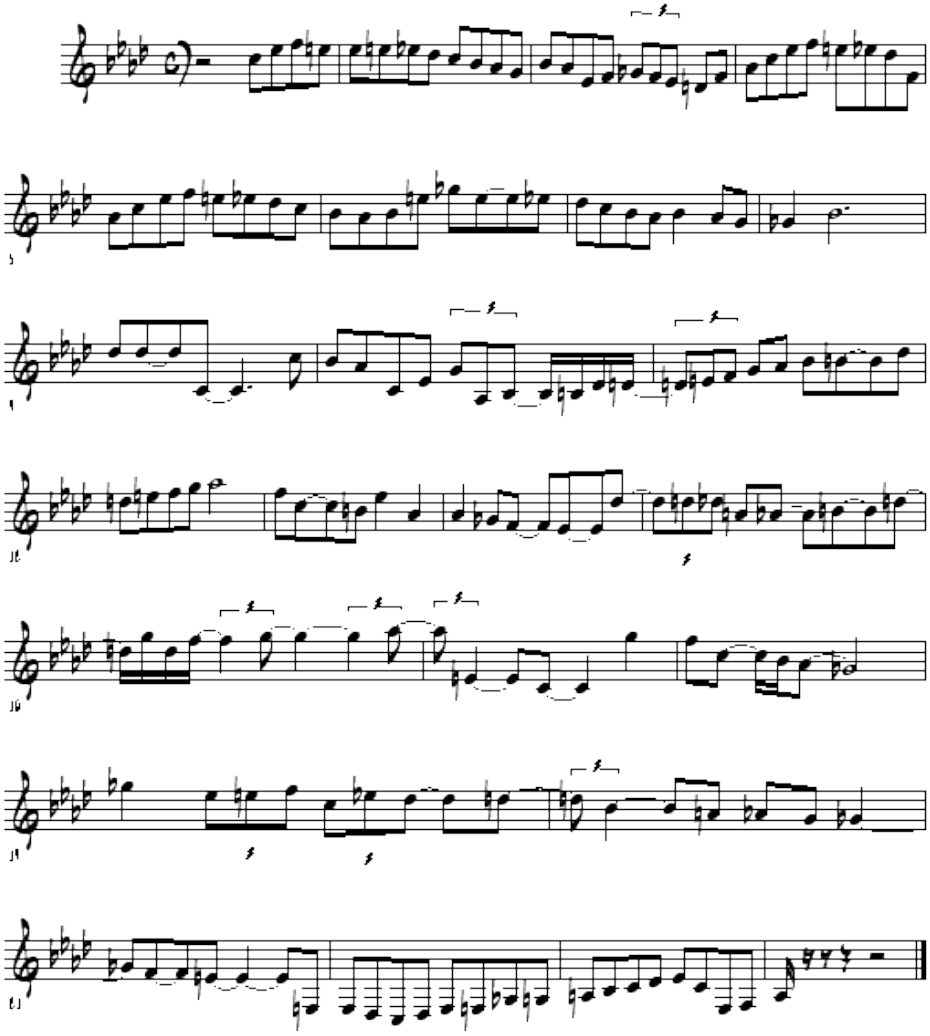
This solution has significant advantages. First, generated music regains much of the diversity, spirit and human appearance in the original piece. Moreover, since it is possible to restrict the analytic information and therefore find more redundancy in the original sequence, the generation phase becomes less constrained. At each generation cycle, every context features many more possible continuations than before.

5.2 Music Composition System Overview

The software was implemented as a user library in an Open Source visual programming language developed at Ircam, called OpenMusic (Assayag & al 1999b). Each step of the algorithm - pre-analysis, learning, generation, and post-generation, is a function, which, in the musical representation software, is represented by a box featuring inlets and outlets. The user may tune up all parameters or add additional operations and constraints by applying additional tools that are available in the OpenMusic system through visual programming.

The process of learning-generation is the following: Once a multi-voice Midi file is transformed into a linear text based on the cross alphabet, it is presented to the learning algorithm. This algorithm produces a prediction tree (also called a motif dictionary). The resulting tree is then randomly traversed by the generation procedure to build variants of the original music.


example1
A midi file.
midi->cross
midi->cross translates the midi file into a sequence of symbols.
lzify
lzify proceeds to the LZ style analysis.
lzgenerate
lzgenerate generates a new sequence from the LZ style analysis.
cross->chordseq
cross->chordseq translates this new sequence into a chord-seq.



6. Musical Experiments and Discussion

Extensive musical experiments have been carried in order to test our system and compare the two generative algorithms. Midi files were gathered from several sources, including polyphonic instrumental and piano music, with styles ranging from early renaissance and baroque music to hard-bop jazz.

In cases where the Midi file was derived from a live recording, the combination of the simplification toolbox and the learning-generation gives excellent results. The simplification is also important in cases where the overall complexity leads to a huge alphabet.

Style Interpolation: An interesting feature of our learning method is style interpolation. When a set of different musical pieces is analyzed simultaneously into a common motifs dictionary (such as by learning a concatenation of two different pieces), the generation will "interpolate" over a space spanned by the separate models. Whenever a common pattern of any length is found to exist between the separate dictionaries during training, the generation phase may create smooth transitions from one style to another by choosing at random a continuation path that belongs to the other model.

Testing over a set of 2-voices Bach Inventions, the results become "infinite" streams containing interesting subsequences, showing original and convincing counterpoint and harmonic patterns, consistent with Bach's style.

IP vs. PST: It is interesting to observe that the two methods, although statistically related, produce at times significantly different musical results. Generally, our impression is that IP tends to capture correctly musically meaningful motifs of the piece, with a tendency to copy complete blocks from an original and "mimic" the original by funny juxtapositions of these blocks, ridiculing the concepts of rhythm, meter and form. The PST on the other hand shows more "inventiveness" by introducing transitions in the middle of a bar or a phrase, thus creating more interesting transitions and not just juxtapositions. The tradeoff is that the PST might cause "false notes" to appear more often.

Below we present a short comparison of IP vs. PST properties that might shed some light on the relationship between statistical and musical criteria:

IP	PST
* Guiding principle: loss-less coding.	* Lossy compression.
* Everything is parsed: this "completeness" assures that all observed rare transitions are included.	* Selective, partial parsing: rare events and events that do not improve prediction are ruled out.
* On-line estimation (instantaneous coding).	* Batch analysis ('file' compression).
* Conditional empirical probability estimation is	* Robust conditional empirical probability

poor for short inputs since the IP parsing is very sensitive to even a single symbol change.	estimation for the significant events.

***Available results*:** The programs that were used to conduct the musical experiments are available as part of OpenMusic, a Lisp-based visual language for music composition (Assayag & al. 1999b). In an accompanying web site we show convincing examples, including a set of piano improvisations in the style of Chick Corea, another one in the style of Chopin Etudes, polyphonic Bach counterpoint, 19th century symphonic music, and modern jazz style improvisations derived from a training set fed by several performers asked to play for the learning system. These results are available at http://www.ircam.fr/equipes/repmus/RMPapers/.

The described system has already served as a tool for computer assisted composition. An original piece by Jean-Remy Guedon, named miniX, has been created recently at Ircam with the assistance of Frederic Voisin, in collaboration with the French "Orchestre National de Jazz". In this 20-minute piece, about half of the solo parts were IP Generated and transcribed on the score to be used in concert.

7. Conclusion and New Directions

In our work, learning is indeed unsupervised since we do not feed our system with any musical knowledge. Rather it analyzes music with constrained algorithms and does not proceed to any inductive inference. Another approach would be to try and induce automatically some conclusions about the presence of musical structures or music schemata, with the help of minimal cognitive mechanisms.

Additionally, we have encountered the question of what determines rareness in music and how one can distinguish between statistically meaningless events vs. musically significant musical material that usually appears near the dramatic climax of the piece. It seems that a different meaning should be assigned to rare events that appear next to the most significant events. Moreover, these rare events should result from some type of a musical operation, such as inversion[1], harmonic[2] or counterpoint[3] change of a significant event. In order to properly deal with this problem we would need to address the schemata estimation issue mentioned above. In our current approach, the use of "smoothing" might eliminate such events from the learned model.

Finally, one may try instead of modeling sequence $x_1, x_2, ..., x_n$ based on its own redundancies, to try and model it based on the redundancies of a second, correlated sequence (e.g., a second voice) $y_1, y_2, ..., y_n$ (through

[1] J.S.Bach, Fugure in C minor, Book I: Near the end of the piece occurs a sudden inversion of a motive that appeared over 20 times in its original form throughout the piece.

[2] L.V.Beethoven, Moonlight Sonata, part I: Contains a strong repetitive harmonic sequence. Near climax a change in harmony occurs that creates a "dissonance" to the melody that remained the same.

[3] J.S.Bach, Fugue in D sharp minor, Book I: Throughout the piece the separate themes appear in a free counterpoint. In the middle of the piece there is an occurrence of all themes in exact concurrence.

the process of transduction). Then, the generation of an X-type sequence could be constrained by an incoming Y-type sequence. This could lead to a synchronous improvisation scheme, where the computer plays alongside with the human performer, keeping an overall polyphonic consistency.

Acknowledgements

We would like to thank the musicologist Dalia Cohen for introducing us to the term “queen of the night” and providing the corresponding musical examples. G.B. is supported by a grant from the Ministry of Science, Israel.

References

Apostolico, A. and G. Bejerano. (2000). "Optimal amnesic probabilistic automata, or, how to learn and classify proteins in linear time and space." *Journal of Computational Biology* 7:381-393.

Assayag, G., S. Dubnov and O. Delerue. (1999a). "Guessing the Composer's Mind : Applying Universal Prediction to Musical Style." *Proceedings of the International Computer Music Conference*. International Computer Music Association, pp. 496-499.

Assayag, G., C. Rueda, M. Laurson, C. Agon and O. Delerue. (1999b). " Computer Assisted Composition at Ircam : PatchWork & OpenMusic.", Computer Music Journal, 23:3.

Bejerano, G. and G. Yona. (2001). "Variations on probabilistic suffix trees: statistical modeling and prediction of protein families." *Bioinformatics* 17:23-43.

Brooks Jr., F., Hopkins Jr., A., Neumann, P., and Wright, W. (1993). An experiment in musical composition, In Schwanauer and Levitt, editors, Machine Models of Music, pages 23– 40. MIT Press.

Conklin, D. and Witten, I. (1995). Multiple viewpoint systems for music prediction, Interface, 24, 51– 73.

Cope, D. (1987) An expert system from computer-assisted composition, in Computer Music Jouranl, 11(4): pp. 30-46.

Cope, D. (1991). Computers and musical style, Oxford University Press.

Dubnov, S., G. Assayag and R. El-Yaniv. (1998). "Universal Classification Applied to Musical Sequences." *Proceedings of the International Computer Music Conference*. International Computer Music Association, pp. 332-340.

Dubnov S., D. Cohen and N. Wagner, (1995), Uncertainty in Distinguishing between Structure and Ornamentation in Music, Proceedings of the International Fuzzy Systems Association Conference, Brazil.

Dubnov, S. and F. Pachet (1998) Technical Report, SONY CSL-Paris, later appeared in F. Pachet. "Surprising Harmonies", International Journal of Computing Anticipatory Systems, (1999).

Ebicoglu, K. (1986), An Expert System for Harmonization of Chorals in the style of J.S.Bach, PhD Dissertation, Department of Computer Science, SUNY at Buffalo.

M.Feder, N.Merhav and M.Gutman, "Universal Prediction of individual sequences", IEEE transactions on Information Theory, vol. 38, pp. 1258-1270, 1992.

Lidov,D. and Gambura, J. (1973), A melody writing algorithm using a formal language model, in Computer Studies in Humanities, 4(3-4), pp. 134-148.

Meyer, L. (1961). Emotion and Meaning in Music. University of Chicago Press, Chicago.

Pinkerton, R. (1956). Information theory and melody. Scientific American, (194), 76– 86.

Ron, D., Y. Singer and N. Tishby. (1996). "The power of amnesia: Learning probabilistic automata with variable memory length." *Machine Learning* 25(2-3):117-149.

Shannon, C., “A mind reading machine”, Bell Laboratories memorandum, (Reprinted in the Collected Papers of Claude Elwood Shannon, IEEE Press, pp. 688-689).

Ziv J, Lempel A, "Compression of individual sequences via variable rate coding", IEEE Trans. Inf. Theory, 24:5, pp.530-536, 1978.