Toward Mechanized Music Pedagogy

Lingfeng Yang*
Stanford University

Abstract

We can adapt existing techniques for the statistical modeling of music to the modeling of the *quality of musical performance*, resulting in a fine-grained statistical understanding of 'hardness' in music performance. This drives AI-assisted applications for improving skill. We present modeling results using various probabilistic models to model performance in the simplified setting of rhythm games.

1 Introduction

Learning a musical instrument is hard. Individually, thousands of dollars in fees are spent taking lessons, and years spent practicing— at a suboptimal level. Practice without constant expert supervision often involves neglecting the real technical shortcomings and worse, learning the wrong habits. The rare state in which we are addressing our shortcomings and learning the right habits is known as deliberate practice [Ericsson et al. 1993]. How can machines help us be in this state more often by automatically pointing out what needs work through the statistical modeling of player performance?

First is the educational value of having the statistics alone. For example, if we know we miss a note sequence particularly often through the statistics, we become better informed of what note sequences to practice. Second is the automation of secondary pedagogical elements such as practice songs; by obtaining the most-likely-to-miss note sequences, we can use techniques from algorithmic music to synthesize a practice song that specifically targets a particular technical weakness. Third, is an objective, detailed metric of improvement; if an accurate statistical model formed for predicting missed note sequences becomes increasingly inaccurate for data the player feeds it in the future, presumably after practicing and getting better, we can tie technical improvement to the lowered empirical probabilities of missed note sequences versus the predicted probability.

In this paper we present how the variable-order Markov model, a model often used to statistically model music, may be used to form similarly detailed models of the quality of musical performance.

2 Related Work

This is an extension of the work in [Yang 2010].

Music cognition. The most related field would be in music cognition. Music cognition is concerned about better understanding how people relate to music. Recently, there has been some interest in modeling music cognition [Honing 2006]. It has not been established yet what "good" models are in this space. Seeing how well our proposed models predict player performance may lead to a better understanding of how "hard" songs are hard and "easy" ones, easy.

Video games and player modeling. The next most related field is in video games. Recently, there have been academic research efforts in modeling players of video games. [Pedersen et al. 2009] discusses how to model fuzzy concepts such as "fun" and "satisfaction" in platform games. There is also work in modeling player performance, but for different game types. [Drachen et al. 2009].

Algorithmic music. Finally, the models we plan on using can also be found in algorithmic music, which is concerned with the modeling and generation of music itself. [Conklin 2003] provides a survey of statistical techniques. [McCormack 1996] gives an overview of music generation techniques. There is also work in using statistical models to predict the names of songs [Brochu and De Freitas 2003]. In particular, we propose to adapt such methods from modeling or generating music to predicting player performance. Broadly, we are interested in gaining a statistical, music-theoretic understanding of skill development in musical performance.

Markov models. Markov models have traditionally been used to obtain detailed statistics of sequential information, including music. The simplest kind of Markov model is the Markov chain, where each member of a sequence of random variables is conditionally dependent on the previous member. Variable-order Markov models (VMMs), the class of model used in this work, are an extension of Markov chains that allow context-specific conditioning on contexts of varying length. We draw our methods heavily from [Begleiter et al. 2004] which gives an overview of the available techinques for training VMMs. Other literature covering VMM training exist, such as [Schulz et al. 2008]. A recent development is *Markov random fields,* which additionally generalize the *shape* of conditioning contexts. In fact, Markov random fields were used recenty in [Lavrenko and Pickens 2003] to model polyphonic music.

3 Problem Domain

In a rhythm game, the player hits notes by pressing buttons on an input device. Each button on the input device corresponds to a note. These input devices tend to be modeled on actual instruments, but the number of notes they encode is much lower; commonly, the number of notes is around 5.

During the course of the game, the player's goal is hit incoming notes in time to the music; i.e., after starting a game, at each point in a sequence of predetermined time intervals, the player has the chance to hit a specific note within a timing window. At high levels of play, this can become very difficult; it is common for a rhythm game to replicate the notes in a technically difficult real musical piece one for one.

We can formalize these concepts as follows. First is the representation of songs in the rhythm game: a song is a sequence of (note, time) pairs:

$$S := \{s_i = (n_i, t_i), n_i \in P, t_i \in \mathbb{R}, t_i \leq t_{i+1}\}_{i=1}^{N},$$

where N is the number of notes in the song and each n_i is from a finite set of *pitches* $P = \{0, 1, \ldots, H\}$. We may represent the player's input as a similar sequence

*e-mail: lyang@cs.stanford.edu

$$I := \{(n_i, t_i), n_i \in N, t_i \leq t_{i+1}\}_{i=1}^{M}.$$

Missed notes as a skill metric. In this setting we define skill as the ability to reproduce the note sequence accurately; i.e., the more notes missed relative to the same song, the lower the skill of the player. However, there are still many different ways in which to measure whether a note is missed.

One way is a timing measurement; for each $s_i = (p_i, t_i) \in I$ such that there is no $s_j = (n_j, t_j) \in S$ within a timing window of t_0 of t_i; $|t_i - t_j| > t_0/2 \quad \forall j$. In general, this is a boolean predicate $P_{S,I} : S \to \{0, 1\}$. Note that this does not capture all the ways of missing a note. in particular, it does not capture the case where the player plays too many notes.

This then leads to a labeled version of S, S_I.

$$S_I := \{(n_i, t_i, P(s_i)), s_i \in S, n_i \in C, t_i \in \mathbb{R}, t_i \leq t_{i+1}\}_{i=1}^{N}.$$

Miss rate. We can then consider S_I as arising from some probability distribution P over all $x_i \in S_I$. We formulate the *miss rate* at each note, the context-specific probability that the player will miss a note, based on this notion:

$$M_i = P((x_i)_3 = 1 | x_1 \dots x_{i-1})$$

A statistically fine grained notion of miss rate is important because it allows for more effective applications for skill development. Note that we do not yet deal with *polyphonic* music, where (n_i, t_i) pairs with the same t_i are allowed. Currently being investigated is adapting the *Markov random field* method for collecting statistics on polyphonic music [Lavrenko and Pickens 2003] to collecting statistics on performance quality taking polyphonics into account.

4 System Overview

To serve as a platform for this research, we have developed a rhythm game that is a spinoff of the Beatmania IIDX [Konami 1999] rhythm game; it augments the regular game with a data collection mechanism. The game is compatible with existing file formats used by the Beatmania modding community; with this, we may test our ideas on a catalog of songs that are already available.

In a typical session, the application may be played like the original Beatmania IIDX; the player starts the game and selects from a menu of songs to play, plays the song, and can optionally play another song. Unlike most rhythm games, however, detailed data is collected per play; After playing each song, the sequences S labeled with missed notes from I and the timestamp of the song along with text delimiting the play session is appended to a log file.

In addition to the data collection are features that represent explorations into how we can apply the results of modeling the data to assisting skill development. We include a facility for generating practice songs that consist of repeats of the most likely note sequences that result in a missed note. As the scope of this paper deals with the modeling aspect, we will not go into it here.

Labeling of data. Chordal and other timing information is then stripped to produce a string with alphabet consisting of consecutive hit/miss symbols $s_i \in \{Hn, Mn\}$ where n is one of P musical pitches:

$$HA|MB|HC|HA|MC|HB|MC|HA|MC \dots$$

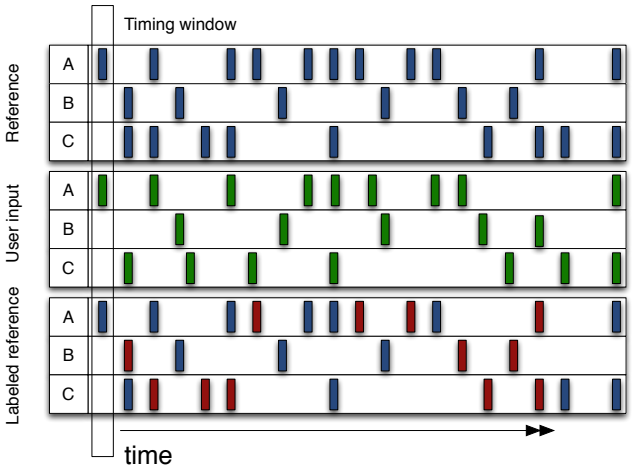

Figure 1: *We take a reference song (blue, top) and user input over that song (green, middle) to obtain a labeling of the reference song with each note as hit (blue) or missed (red) (bottom). Here we give an example for 3 pitches $\{A, B, C\}$. We use 7 pitches in our actual system.*

The miss rate at note i may then equivalently be calculated as

$$M_i = P(s_i = Mn | pitch(s_i) = n, s_1 \dots s_{i-1}).$$

5 Modeling

In the setting of mechanizing music pedagogy, there are three aspects we require of any model that can realize the probability query above.

1. The model makes accurate predictions. This is clear; the query given above should be able to predict what notes the player will get wrong.

2. The model generalizes. The query given above should be able to perform these predictions on data that is not from the training set.

3. The model is resilient to data starvation. Because we target individual players, one should not be forced to generate a vast dataset of performances before the model is usable. The query should give accurate results from a single player on a single playthrough of a single song.

An established method for modeling probabilities over strings is the *Markov chain* of order k, also known as the k-gram model. Each consecutive occurence of k symbols in the string is matched against the symbol that comes next. By accumulating a list of (k-sequence, next symbol) pairs, one also accumulates the conditional probability

$$P(s_i | s_{i-1} \dots s_{i-k}).$$

These models can be very accurate. However, in this case we are interested in the contextual miss rate from the *beginning of the entire song.* If we were to use fixed-order Markov models, this would require multiple models of very high order (for a typical song, in the hundreds). This would require

an enormous amount of performance data from *each* player; as this means the model is not resilient to data starvation we cannot use a k-gram model for our purposes.

6 Variable-Order Markov Models

Because fixed-order models cannot be applied practically to our modeling problem, we turn to variable-order Markov models (VMMs). VMMs are an extension of fixed-order Markov models wherein one is allowed to condition on any context of arbitrary length, not just those of a fixed length, as is the case in fixed-order Markov (n-gram) models.

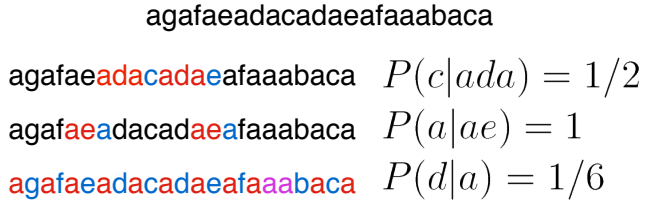

Figure 2: *A possible VMM for a string.*

Allowing conditioning on arbitrary-length contexts lets one concentrate the probability mass to contexts of different lengths, potentially resulting in much better accuracy and generalization while requiring much less data.

6.1 Training the VMM

Several algorithms exist for learning variable-order Markov models. Essentially what needs to be done to learn a VMM is to pick a context to match to each symbol in the alphabet, and then learn the probabilities using these (symbol, context) pairs. The variation is in how contexts are selected.

To gain high accuracy and generalization, good VMM learning algorithms generally learn contexts that maximize the likelihood of the training sequence $x_1^T = x_1 \dots X_T$:

$$\hat{P}(x_1^T) = \prod_{i=1}^{T} \hat{P}(x_i | x_1 \dots x_{i-1}).$$

The likelihood is taken to be the joint probability of all symbols in the sequence occuring given all previous symbols.

It is not trivial to determine which contexts will do this given finite computational resources. Several algorithms select these contexts based on information-theoretic principles; in fact, any lossless compression algorithm that is based on storing a dictionary of phrases additionally induces a VMM over the input sequence [Begleiter et al. 2004].

The LZ-78 algorithm. One such compression algorithm is the LZ-78 algorithm, which compresses well (acheives high likelihood) while also being extremely fast. The LZ-78 algorithm works as follows (see Figure 2): A string is consumed from beginning to end, building up a phrase dictionary. The shortest phrase not in the dictionary is parsed and inserted into the dictionary at each step.

The LZ-MS algorithm. The LZ-MS algorithm is a variant. It has two parameters, M and S. LZ-MS performs the LZ-78 algorithm on $S \geq 0$ 1-symbol *shifts* of the input string and *backtracks* by M symbols each time a phrase is parsed, accumulating a more 'complete' dictionary in the process. Further details are available in [Begleiter et al. 2004]. We employ the LZ-MS variant in our VMM learning algorithm.

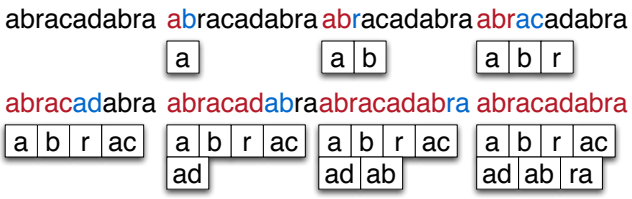

Figure 3: *The LZ-78 algorithm applied to the string 'abracadabra.'*

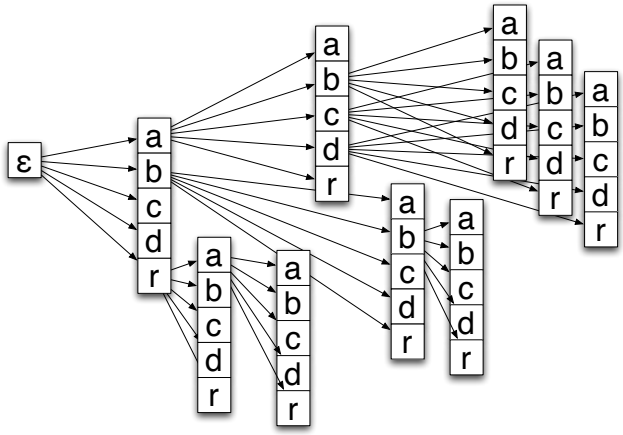

Figure 4: *The decision tree corresponding to the phrase dictionary of 'abracadabra.'*

From phrase dictionary to decision tree. The LZ-78 algorithm produces a set of phrases given an input string. There is a corresponding tree from which conditional queries of any context length may easily be calculated. We may consider the phrase set as being generated by taking nondeterministic walks in the tree up to one level before the leaves. Further details of how this is works is available in [Begleiter et al. 2004].

Consider the phrases $\{a, b, r, ac, ad, ab, ra\}$. The first character in each of the phrases must be from the alphabet $\{a, b, c, d, r\}$, hence the first level in the tree is $\{a, b, c, d, r\}$. Next, we consider the second character *given* each of the first characters; given a, it can be $\{c, d, b\}$, and given b, there are no choices, and given r, it can be $\{a\}$. We proceed in a similar fashion until all the phrases in the dictionary are exhausted, then add another alphabet $\{a, b, c, d, r\}$ to the tree. An example of a conditional probability calcuated with walks in the tree:

$$Cond(n) = c(n) / \sum_{s \in sibs(n) \cup \{n\}} c(s)$$

$$P(b|ar) = Cond(\epsilon \to a \to r \to \epsilon \to b) = 5/33$$

7 Results

A test of the LZ-MS-based VMM's prediction accuracy and generalization was run on a single playthrough of a single song. The resulting labeled string comprised 700 symbols in length. The VMM was trained on 600 symbols with a contiguous section of 100 symbols left out to serve as the test set.

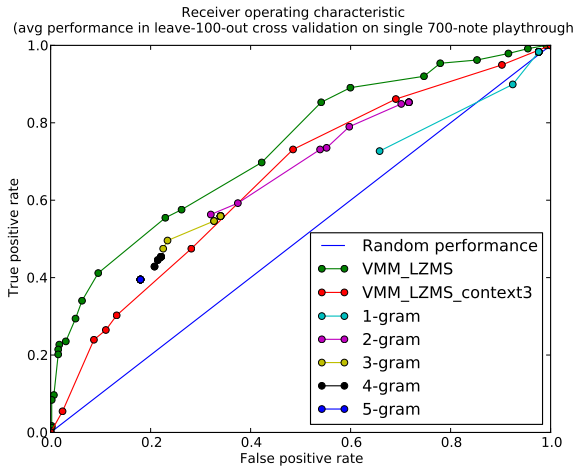

Figure 5: *The receiver operating characteristic of the LZ-MS VMM model on the prediction query* $P(x_i = Mn|pitch(x_i) = n, x_1 \dots x_{i-1})$. *Note the inferior performance of both the LZ-MS VMM model and 1-to-5-gram models on the simplified fixed-order-k prediction queries* $P(x_i = Mn|pitch(x_i) = n, x_{i-1} \dots x_{i-k})$.

Receiver operating characteristic. In Figure 5 we see that the algorithm achieves better than random performance in its *receiver operating characteristic*, which is obtained by using the predicted probability $\hat{p}$ along with some threshold probability $\theta \in [0, 1]$ where if $\hat{p} > \theta$, we decide that the note was missed, and if the note actually is some Mn, it is a true positive, else it is a false positive. The receiver operating characteristic is the curve obtained by plotting true/false positive performance over several $\theta \in [0, 1]$. The same evaluation metric is used in [Lavrenko and Pickens 2003]. We believe a similar metric is appropriate for predicting missed notes as it is for note occurences in general.

Comparison with other models. There are two decisions we made in our setting that demand comparison: one is the choice of VMMs over fixed-order Markov models. Another is the choice of conditioning starting from the very beginning of the song

$$P(x_i = Mn|pitch(x_i) = n, x_1 \dots x_{i-1})$$

which is a variable-order query, versus on some fixed-order context

$$P(x_i = Mn|pitch(x_i) = n, x_{i-1} \dots x_{i-k}).$$

In Figure 5, we see that prediction accuracy and generalization ability of the LZ-MS VMM algorithm is crippled when given the fixed-size query, while fixed-order models do better with the fixed-size query, they do not do as well as the LZ-MS VMM algorithm on the variable-order query. Finally, the ability of the variable-order query to predict note misses is demonstrated to be superior to that of the fixed-size query, at least with these particular realizations (VMMs and fixed-order Markov models).

We do not include fixed-order model results on the more variable-order query because it requires the learning of an order-at-least-100 Markov model, which in this setting would clearly not work due to data starvation; running such a model on our training set resulted in uniform answers, as the model learned a very low probability of seeing *any* note conditioned on a length-100 context.

Certainly other alternatives exist, like smoothed combinations of fixed-order models and alternative ways of learning VMMs. They are not included because we consider these essentially VMMs, and our comparison is between VMMs and non-V MMs, not among VMMs.

Performance. The faster performing a learning algorithm is, the better it can be used in a real-time application such as rhythm games. It is better to get feedback in form of an updated distribution quickly after playing each song. For the 700-note training set referenced in Figure 5, the learning algorithm constructed the phrase dictionary and decision tree on the order of tenths of seconds on an 2.66 GHz Intel Xeon (using 1 core). Miss-rate queries approached real-time performance (on the order of hundredths of seconds). This performance profile allows the learning algorithm to be used for every song that the player plays, and the prediction algorithm for individual notes in a playthrough. The learning and prediction algorithms were implemented in Haskell with little regard to space leaks from lazy evaluation and minimal memoization; even higher performance should be in reach through program optimizations.

8 Limitations and Future Work

Augmentations to the system. One natural next step is to push the software out to more users and have the collected data be available in a cloud-services-like fashion. This would enable the collection of data at a large scale, in turn enabling the usage of learning algorithms that require much more data but can potentially be much more descriptive. In particular, it would enable a player to compare performance versus other players, and to see how other players improved their technique, in terms of how the learned statistics changed for another player based on what songs she played.

Markov Random Fields. Besides not being able to deal with polyphonicity, the criteria of VMMs that a context be contiguous is quite limiting. There is potential to use a better model. We are currently evaluating Markov Random Fields using the method of [Lavrenko and Pickens 2003]. The drawback of this method are that it is much slower than using VMMs with LZ-MS; it could only be used in an offline manner. However, it may prove to be more accurate. The features learned by the method may also be qualitatively insightful to the musician interested in improving technique.

Interactive machine learning. A potentially useful augmentation to any context-based probability model, which relates to interactive machine learning, is to let the contexts be selectable by the user, fixing a set of them, and then selecting the rest based on some other algorithm. This is because often the user has a pretty good idea of what kinds of musical contexts they make mistakes in.

Other applications. Currently considered applications include the automatic synthesis of 'hard enough' practice songs and tools to track technical progress at a fine-grained level. What other applications are there?

Finally, the general area of AI-assisted human learning is a promising research direction. We've spent a lot of time teaching computers how to do things better— so much that the computers might as well be teaching us.

References

Begleiter, R., El-Yaniv, R., and Yona, G. 2004. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research 22*, 1, 385–421.

Brochu, E., and De Freitas, N. 2003. " Name That Song!" A Probabilistic Approach to Querying on Music and Text. *Advances in neural information processing systems*, 1529–1536.

Conklin, D. 2003. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, Citeseer, 30–35.

Drachen, A., Canossa, A., and Yannakakis, G. 2009. Player modeling using self-organization in tomb raider: underworld. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG2009), Milano, Italy. http://www. itu. dk/˜ yannakakis/CIG09_IOI. pdf.*

Ericsson, K., Krampe, R., and Tesch-R "omer, C. 1993. The role of deliberate practice in the acquisition of expert performance. *PSYCHOLOGICAL REVIEW-NEW YORK- 100*, 363–363.

Honing, H. 2006. Computational modeling of music cognition: A case study on model selection. *Music Perception 23*, 5, 365–376.

Konami. 1999. Beatmania IIDX.

Lavrenko, V., and Pickens, J. 2003. Polyphonic music modeling with random fields. In *Proceedings of the eleventh ACM international conference on Multimedia*, ACM, 120–129.

McCormack, J. 1996. Grammar based music composition. *Complex systems 96*, 321–336.

Pedersen, C., Togelius, J., and Yannakakis, G. 2009. Modeling player experience in super mario bros. In *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009*, 132–139.

Schulz, M., Weese, D., Rausch, T., D "oring, A., Reinert, K., and Vingron, M. 2008. Fast and adaptive variable order markov chain construction. *Algorithms in Bioinformatics*, 306–317.

Yang, L. 2010. Modeling Player Performance in Rhythm Games. *To appear in: SIGGRAPH Asia 2010 technical sketches*.