Machine Learning of Expressive Microtiming in Brazilian and Reggae Drumming

Matt Wright (Music) and Edgar Berdahl (EE), CS229, 16 December 2005

Abstract

We have used supervised machine learning to apply *microtiming* to music specified only in terms of exact, quantized, theoretical note times. Specifically, the input consists of a sort of musical score: note data specifying the quantized attack times for a variety of percussion instruments. The output of the regression schemes we tried is simply the microtiming deviation to apply to each note. In particular, we trained Locally Weighted Linear Regression / K-Nearest-Neighbors (LWLR/KNN), Kernel Ridge Regression (KRR), and Gaussian Process Regression (GPR) on data from skilled human performance of a variety of reggae drum parts and Brazilian rhythms. Evaluating our results with cross-validation, we found that the three methods are quite comparable, and in all cases the mean squared error is substantially less than the mean squared microtiming of the original data. Subjectively, our results are satisfactory; the applied microtiming captures some element of musical style and sounds much more expressive than the quantized input.

Introduction

Most styles of music are based on a theoretical model of rhythm in which the timing of notes is specified in terms of exact integer divisions of the beat. In real performance by skilled human musicians, a large part of the expression comes from *microtiming*, small but meaningful variations in the exact timing of notes to produce "feel," "groove," or "swing." Computer-generated realizations using perfectly-timed notes lack microtiming and are generally characterized as "mechanical," "dry," "inhuman," etc.

We applied supervised machine learning to the problem of generating stylistically appropriate microtiming for notes specified only in terms of theoretical exact times, i.e., notes whose onset times are exactly submultiples of the beat and that therefore have zero microtiming. In particular, our algorithm's input is the MIDI score of a series of drum notes, and the output is the microtiming to apply to each note. MIDI is a symbolic representation that (for our purposes) represents music by specifying timing, loudness, and timbre (i.e., which drum) for each note.

A musician can manipulate other parameters of a performance in order to influence its expressiveness, including tempo, dynamics, and articulation. Currently we address only microtiming: tempo is constant for our training data (because it is almost constant in these musical styles); dynamics and articulation we leave for future work.

Related Work

Most of the related work pertains to synthesizing expressive concert piano music. The oldest line of research of which we are aware began in 1983 at the Royal Institute of Technology (KTH) in Stockholm, where Johan Sundberg and others determined rules that should govern the expressive parameters of music[4]. They used an iterative process by inventing rules, asking professional musicians to comment on the musical output of the algorithms governed by the rules, then manually fine-tuning the rules. The rules were therefore limited by the imaginations of the researchers, and the parameters for the rules were approximated using a limited number of listening tests. Nevertheless, many of the rules were able to predict performance parameters in tests on actual performance data from Mozart sonatas provided by Gerhard Widmer [5]. However, some rules systematically failed on certain musical phrases, leading to the conjecture that such rules should incorporate knowledge of the phrase structure of a piece, at least for the domain of concert piano.

In contast, Gerhard Widmer sought to use machine learning to automatically "discover" the rules governing expressive concert piano music performance, and to have these rules expressible in simple English. In his first attempt, he applied nearest-neighbor methods, Bayesian classifiers, decision tables, decision trees, and classification rules to the melodies of thirteen sonatas by Mozart, concluding that decision trees and classification rules performed best [7]. He considered only the discretized and therefore drastically simplified classification problem, i.e., predicting whether a given note would be played early or late, with resulting accuracies in the neighborhood of 60%. He further determined that different models were needed for different meters and tempi and that rhythmic structure was more closely tied to tempo deviations, while melodic structure had more influence on the dynamics.

Widmer later became interested in the idea of "partial models," which he desired to explain some characteristics of expressive performance without causing overfitting due to the learning algorithms attempting to explain every training sample. As a result, Widmer developed the PLCG ("**p**artition, **l**earn, **c**luster, and **g**eneralize") algorithm, based on a sequential set covering algorithm [2] and some generalization heuristics. Many of the resulting

expressive music performance rules coincided with musicological theory [8].

Next Widmer added some higher-level features to further aid his learning algorithms. However, the features were so high-level that they required a manual and somewhat subjective musical analysis of the training data by music experts. The phrase information was deemed the most important aspect of these features. In particular, each note was considered to belong to phrases of similarly behaving notes on both smaller and larger timescales. In essence, the learning algorithms were using human input to help cluster perceptually related notes together [6].

The hierarchical nature of the new features allowed Widmer to experiment with some more exotic inductive learning algorithms such as the relational case-based learner DISTALL. It used a distance measure which when applied to two phrases, not only measured the dissimilarity between the two phrases A and B, but also measured the dissimilarity between the phrases *related* to A and B, where the relatedness between phrases was determined using the high-level phrase analysis features [6].

While this method indeed shows promise, it has some weaknesses because of which we have decided not to implement it. In particular, Widmer states that he has had better results with the dynamics than the timing and tempo, whereas we are most interested in the timing. In addition, Widmer makes some assumptions about how the local dynamics and tempo of each phrase contribute to the overall dynamics and tempo. That is, he assumes that they are additive and makes no musicological argument for why this should be the case. He further chooses to parameterize the local dynamics and tempo curves using second-order polynomials, and this decision seems motivated more computationally than musicologically. Finally, Widmer even admits that the largest source of error in the model was the fitting of quadratic functions to the dynamics and tempo curves [9].

Training Data

We have purchased collections of MIDI sequences of drum parts for Reggae and Brazilian music genres from KEYFAX NewMedia. Expert percussionists in each genre recorded these sequences in real-time via MIDI-equipped acoustic drum sets and similar input devices. In general each consists of about 16 bars of a basic rhythm with minor variations, played on the full battery of instruments. Subjectively, these sequences "sound authentic": their microtiming variations are enough to give a sense of each musical style. We manually selected sequences that were long enough (at least 8 bars), sounded "authentic" to us, and had some degree of interesting microtiming, resulting in the following collection of training data:

Rhythm	# notes	# insts	Mean|MT|	Std(MT)
Escola	1189	10	0.0282	0.0346
Olodum	1079	13	0.0261	0.0248
Sambareg	904	10	0.0181	0.0235
Rokbahia	522	5	0.0168	0.0217
Maracana	1043	8	0.0345	0.0378
Partalto	739	10	0.0219	0.0255
Sambafnk	960	11	0.0375	0.0315
Afoxe	556	7	0.0187	0.0229
Baiao	1020	9	0.0240	0.0307
Reggae	897	22	0.0188	0.0248
Lvr-rock	895	17	0.0043	0.0102
Ska	382	6	0.0216	0.0293

Table 1: Input data by rhythm, showing number of notes and instruments, and mean absolute value and variance of microtiming (MT, measured in beats)

Since the input to our machine learning algorithm is the unexpressive, "perfectly" timed version of a rhythm, we had to *quantize* these expressively timed sequences. In some respects this is analogous to transcribing a rhythm in written notation, because it involves making subjective categorical judgments about the rhythmic "meaning" of each note according to the presumed underlying metrical structure. That is to say, though the quantization process itself is purely mechanical and deterministic, the choice of which level of subdivision to quantize to involves human subjectivity.

Quantization actually turned out to be much more difficult than we anticipated, because many of the parts had some notes at the 32nd-note level, yet many of the notes that we considered to be 16th notes have such large microtiming (i.e., they were played so early or so late) that they were assigned to the wrong beat when quantized to a 32nd-note grid. We solved this problem with a heuristic hybrid quantization algorithm that quantizes each note to both the 16th-note and 32nd-note grids, preferring the 16th-note result unless multiple notes from the same timbre "collide" by being quantized to the same 16th note, in which case the algorithm then uses the 32nd-note quantization. In other words, this heuristic makes the strong modeling assumption that all notes at the 32nd-note level are part of an adjacent pair of 32nd notes, not part of a syncopation.

Distance Metric

All of the learning algorithms we tried, with the exception of GPR, are based directly on some notion of *distance* between a given input note and each note in the training data. Our goal in devising the distance metric was to account for the factors that we believed determine how a given note will be microtimed:

timbre, position in the rhythmic cycle, and the presence of other notes nearby in time. Our distance metric has three components:

- Timbral distance is zero between two notes on the same timbre (e.g., snare drum), one between notes of totally different timbres (e.g., crash cymbal vs. bass drum) and a subjective ad hoc number between zero and one for instruments that sound similar and/or have similar rhythmic functions.
- Metric position distance is zero between two notes at the same point in a bar (e.g., both notes on the 16th note after beat two), small for notes in positions that perform similar rhythmic functions (e.g., the 8th note after beat one vs. the 8th note after beat three), and one for notes at rhythmically dissimilar positions.
- Rhythmic context difference captures the effect on microtiming of notes played shortly before or after the given note on the same or different timbres. Our current implementation looks at the timbres and relative temporal position of all notes within one beat of the input notes, although we could certainly extend this window if it proves helpful.

Our current distance metric is nonnegative and symmetric, but the triangle equality does not necessarily hold.

K–Nearest–Neighbors / Locally Weighted Linear Regression

The simplest algorithm we tried outputs a linear combination of the microtimings for the k training notes "nearest" to the given test note, weighted weighted (inversely) by distance. For example, we compute the weight $w^{(i,j)}$ corresponding to the distance between the notes i and j as follows.

$$w^{(i,j)} = \exp\left(-\frac{\text{dist}(i,j)}{2\tau^2}\right)$$

When k equals the size of the training set, K-Nearest-Neighbors (KNN) is equivalent to Locally Weighted Linear Regression (LWLR); this was never the optimal value of k.

We selected k and τ to minimize the mean squared error (MSE) over all our data with 5-fold cross-validation, resulting in k=26 and τ=24. Selecting k and τ to minimize the MSE for each rhythm results in only a very small decrease in test error, so we believe the k and τ parameters to be global.

Kernel Ridge Regression

We realized that KNN was not using knowledge of the actual microtimings in deciding which training set elements to use for prediction, so we applied Kernel Ridge Regression (KRR) to try to improve the performance. This formulation of a general regression problem using kernels was very useful for us because it allowed us to directly apply our distance metric, which was motivated by our intuition into this particular problem.

While our distance measure is nonnegative and symmetric, the triangle equality does not necessarily hold. Furthermore, our distance measure is not a Mercer kernel, and so the matrix K of pair-wise distances $w^{(i,j)}$ is not positive semidefinite in general. Therefore, to ensure that we were able to backsolve the necessary set of linear equations, we needed to pick the magnitude of the ridge term λ to be at least slightly larger than the smallest eigenvalue of K. We used cross-validation to determine the optimal values for λ and τ, which suggested that λ should be chosen as small as possible. Luckily, the smallest possible λ was always a few orders of magnitude smaller than the large eigenvalues of K.

Figure 1 shows part of the results from cross-validation for the rhythm *Escola*. For a given note index, a positive stem means that a note was (blue) or is estimated to be (red) played late, whereas a negative stem corresponds to an early note relative to the score. Despite the difficult nature of the regression problem, KRR is able to make reasonable estimates of many of the microtimings. A stem plot of the results from KNN also looks similar.

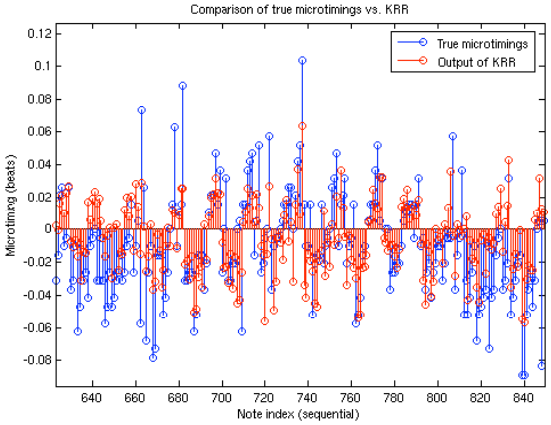


Figure 1: Comparison of the true microtimings versus those predicted by KRR

KRR sometimes performed slightly better than KNN and sometimes slightly worse (see Fig. 2). Perhaps the suboptimal performance was related to the fact that our distance metric was not strictly speaking a kernel. However, we could certainly improve the performance of both KNN and KRR by refining our distance metric as a few of the weights currently used in the metric were chosen by making educated guesses. However, searching this space would be very time-consuming because we would have to use cross-validation, and we might not gain much insight into the problem through this approach.

Gaussian Processes Regression

The typical formulation of Gaussian Process Regression (GPR) is similar to KRR but it allows automatic fine-tuning of the weights involved in estimating distances [3]. We found this aspect of GPR attractive, and so we maximized the log-likelihood with respect to these weights, often called *hyperparameters,* using a conjugate gradient descent method[1].

The main challenge lay in reformulating our distance metric in terms of a Euclidean distance between feature vectors. This was particularly difficult because the easiest way of defining a distance metric (as we did above) is as a function of two notes and their respective contexts in the score, but any feature vector representation needs each note to be translated into an element in a vector. We thus required feature vectors containing several hundred elements in order to incorporate all of the aspects of our distance metric. We formulated our feature vectors for the i^{th} note as follows.

- The first five elements described the i^{th} note's onset time within the measure. We chose these descriptions such that the distances between rhythmically similar positions in the measure would be small compared with distances between other positions in the measure.
- The sixth element describes the "timbral height" of the i^{th} note. That is, timbres with more energy at lower frequencies are assigned smaller timbral heights than timbres with more energy at higher frequencies.
- The seventh element describes the amplitude with which the i^{th} note should be played. In MIDI terminology this is the *velocity* of the note.
- The remaining 595 elements consist of a locally-windowed representation of an exploded representation of the score. The explosion results from forming the cross product of the 35 possible timbres versus the possible note locations in a window surrounding the i^{th} note. Currently we use a window consisting of a beat before and a beat after the i^{th} note. Since we are using a 32^{nd}-note level quantization, there are 17 such positions.

However, we found that given the amount of data we had quantized (see Table 1), we could estimate on the order of only 10 to 20 hyperparameters robustly. Thus, we needed a way to reduce the dimensionality of the score component of the feature vectors. This step was further required due to the computational intensiveness of the conjugate gradient descent method.

Dimensionality Reduction

Principal Components Analysis (PCA) made for a good starting point for this step because of the sparse nature of the feature vectors. In fact, some of the dimensions of the exploded score were completely unnecessary for particular rhythms. For example, rhythms not containing a particular timbre needed to have the corresponding columns in the feature vectors removed, and the preprocessing inherent in implementing PCA could perform this function automatically.

Some work had already been carried out in this area. A paper by Ellis and Arroyo outlines a method for analyzing popular music using what they term "eigenrhythms" and "indirhythms" [1]. They make many genre-specific assumptions that we cannot make, and so we have had to carry out some additional steps to achieve a reasonable singular value spread. These steps require intuition into the important patterns in the data. For example, we found that we needed to apply a rotation of the feature vectors that re-aligns every two beats in a bar with the edges of feature vector. This leads to significantly larger correlations in the data due to the periodic characteristics of rhythms that are not genre-specific.

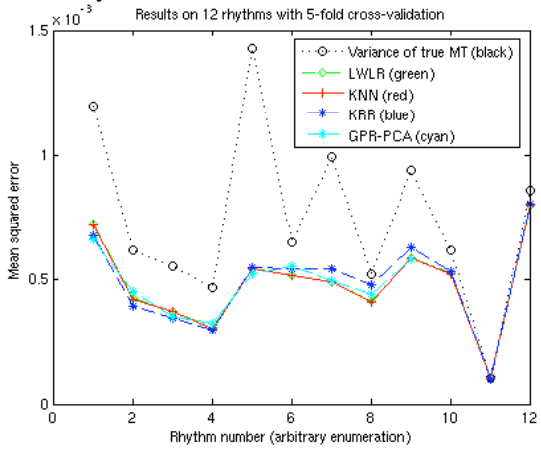
Thus, we used PCA to project the 595 exploded score dimensions down to six. Even though this projection only preserved only about 9% of the variance, it allowed us to apply GPR successfully while using some information from the score. Including the other seven features to which we did not apply PCA, we thus used GPR to estimate thirteen hyperparameters. Judging by the resulting MSE, the GPR method performed approximately as well as KNN and KRR.

[1] MATLAB code released by Carl Rasmussen (2005), http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/

However, we believe that GPR has much more room for improvement. We would like to experiment with a number of additional pre-PCA rotations of the exploded score data, and we also have not yet used cross-validation to find the optimal number of PCA dimensions. In addition, there may be better ways of reducing the dimension of the exploded score that result in eigenvectors more relevant to the regression task. Along these lines, we have carried out some preliminary experiments with independent components analysis (ICA). In particular, we applied ICA to the five components output by PCA. ICA tended to result in more-differentiated components than those generated by PCA. This differentiation is important due to the assumption that the covariance matrix is diagonal.

Results and Future Applications

The performance of all four methods, measured by the MSE on 5-fold cross-validation, is strikingly similar (see Fig. 2). Some rhythms appear more difficult to microtime using our algorithms, but this variation in the MSE is correlated with the variation in the actual microtiming variance of the rhythms themselves. That is to say, some rhythms simply have more microtiming than others. All of our algorithms perform significantly better than applying no microtiming at all, in an MSE sense. In addition, they are also better in a perceptual sense. According to our preliminary and informal listening tests, a machine-microtimed score sounds much more human and less mechanical than a completely regular, quantized score. We of course also made comparisons with randomly microtimed scores where the standard deviation of the random noise applied was the same as that of the actual microtimings applied by a musician. These "random" examples did not sound realistic at all because they were not microtimed consistently.



In the future, besides refining the working details of the GPR implementation to improve the MSE, we plan to work on applying this machine microtiming method. One intriguing application is cross-synthesis, where the microtiming deviations from one rhythm are applied to another quantized score.

References

[1] D. P. W. Ellis and J. Arroyo, *Eigenrhythms: Drum pattern basis sets for classification and generation*, *Int. Symp. on Music Info. Retr. ISMIR-04*, Barcelona, 2004.

[2] J. Fürnkranz, *Separate-and-Conquer Rule Learning*, Artificial Intelligence Review, 13 (1999), pp. 3–54.

[3] C. E. Rasmussen, *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression*, *PhD Thesis, Dept. Computer Science*, University of Toronto, 1996.

[4] J. Sundberg, A. Askenfelt and L. Frydén, *Musical performance. A synthesis-by-rule approach*, Computer Music Journal, 7 (1983), pp. 37–43.

[5] J. Sundberg, A. Friberg and R. Bresin, *Attempts to Reproduce a Pianist's Expressive Timing with Director Musices Performance Rules*, Journal of New Music Research, 32 (2003), pp. 317-325.

[6] A. Tobudic and G. Widmer, *Case-based Relational Learning of Expressive Phrasing in Classical Music*, *Advances in Case-Based Reasoning. 7th European Conference, ECCBR 2004*, Madrid, Spain, 2004, pp. 419-33.

[7] G. Widmer, *Large-scale Induction of Expressive Performance Rules: First Quantitative Results*, *International Computer Music Conference*, Berlin, Germany, 2000.

[8] G. Widmer, *Machine discoveries: A few simple, robust local expression principles*, J. New Music Res., 31 (2002), pp. 37–50.

[9] G. Widmer and W. Goebl, *Computational Models of Expressive Music Performance: The State of the Art*, Journal of New Music Research, 33 (2004), pp. 203–216.