Towards Machine Learning of Expressive Microtiming in Brazilian Drumming

Matthew Wright[*] and Edgar Berdahl[†]

[*]CCRMA (Stanford) and CNMAT (UC Berkeley), matt@{ccrma.Stanford,cnmat.Berkeley}.edu
[†]Department of Electrical Engineering, Stanford University, eberdahl@Stanford.edu

Abstract

We have used supervised machine learning to apply microtiming to music specified only in terms of quantized note 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 Brazilian rhythms. Although our results are still far from the dream of inputting an arbitrary score and having the result sound as if expert human performers played it in the appropriate musical style, we believe we are on the right track. 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.

1 Introduction and Related Work

Most music is 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 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" (Bilmes, 1993; Clarke, 1999; Iyer, 2002). Computer-generated realizations using "perfectly" timed (i.e. quantized) notes lack microtiming and are generally characterized as "mechanical," "dry," "inhuman," etc. Most researchers in music perception and production seem to agree that microtiming is systematic, in other words, that deviations from strict metronomicity are not just random "motor noise" caused by human imperfection (though that is clearly one component), but instead relate to musical structure.

For example, expressive timing of Western classical music has been studied extensively, particularly concert piano music (Sundberg, Askenfelt, & Frydén, 1983), and it is well known that performers in this style utilize local tempo variations to highlight (multiple levels of) phrase structure. The present work, however, deals with dance-oriented musical styles in which tempo remains essentially constant, so models of rubato, ritard, tempo curves, time maps, etc. (Honing, 2001; Jaffe, 1986) are of little use.

Another thrust of rhythm research centers around models in which each note's microtiming is determined by its metrical position. This concept is variously known as "per-measure tatum deviations" (Bilmes, 1993), jazz "swing ratios" (Friberg & Sundström, 2002; Waadeland, 2001), the "composer's pulse" (Clynes, 1983), or, in the commercial world, "groove templates" (Busse, 2002), which are a musically powerful first approximation to stylistically appropriate microtiming. They have the advantage of being easy to implement: to make a groove template from some set of training notes, simply find the mean deviation for all of the notes at each metrical position in the bar. To apply the groove template to a set of quantized test notes, simply offset each test note's timing as a function of its position in the bar. A major disadvantage of groove templates is that they do not capture any effect of context.

Many researchers have applied machine learning (ML) to questions of rhythm and microtiming. In the music information retrieval community, the application is usually musical genre classification. We are aware of far fewer applications of ML for our goal, synthesizing expressive-sounding performance. Again, most research relates to rubato in Western classical styles, particularly concert piano (Widmer, 2000). Bilmes' work (Bilmes, 1993), which is described below in section 3.2, and Grachten's work (Grachten, 2006) are notable exceptions.

2 Our Data

We have purchased collections of MIDI sequences of drum parts for Brazilian music genres from KEYFAX NewMedia.[1] An expert percussionist in these genres recorded the sequences in real-time via MIDI-equipped acoustic drum sets and similar input devices. Each sequence consists of about 16 bars of a basic rhythm with minor variations, played on the full battery of instruments, with totally constant tempo (presumably played to a click track). Subjectively, these sequences "sound authentic": their microtiming variations give a sense of each musical

[1] http://www.keyfax.com/keyfax/prdct.html

style. We manually selected sequences that were at least 8 bars in duration and had interesting microtiming, resulting in the moderately sized training set summarized in Table 1. Since the input to our ML algorithm is a rhythm with no microtiming, we *quantized* each sequence and then calculated the variance of microtiming (the average squared microtiming per note).

Sequence	# notes	# insts	Variance
Escola	1189	10	0.001197
Olodum	1079	13	0.000615
Sambareg	904	10	0.000552
Rokbahia	522	5	0.000471
Maracana	1043	8	0.001429
Partalto	739	10	0.000650
Sambafnk	960	11	0.000992
Afoxe	556	7	0.000524
Baiao	1020	9	0.000942

Table 1: Number of notes and instruments, and variance of microtiming (measured in beats^2) per rhythm

3 What Structures Microtiming?

Machine learning seems to promise that it will automatically find the structure in the training data. In fact, ML works much better if the researcher already understands the structure of the problem enough to preprocess the input to the ML algorithm into a form that exposes the relevant information in the data that one might have hoped the algorithm would learn for itself. For example, Widmer's best results in synthesizing expressive piano performances rely on manual marking of musical phrase structure (Widmer, 2002). Another challenge is the lack of a general way of proving that a given set of data does not contain useful, learnable information. One may only prove the opposite by human inspection in specific cases or by successfully applying a particular learning algorithm.

3.1 How Much of the Microtiming can be Explained by Metrical Position?

We constructed groove templates for each rhythm in our dataset to see how well they explain the data from a mean squared error (MSE) point of view. The mean squared microtiming for all of the notes in our database is 0.00087 beats^2, while the mean squared error between each note's true microtiming and the microtiming predicted by the appropriate groove template is 0.00062 beats^2. This means that groove templates correctly predict just under 30% of the microtiming variance. With an individual groove template for each instrument, the result was just under 39%. Clearly groove templates do not tell the whole story.

3.2 What Else Structures Microtiming?

Little prior work analyzes the musical parameters that correlate to or explain microtiming in the constant-tempo case. (Bilmes, 1993) uses an ML approach to segment improvised drum rhythms into phrases and cluster them by similarity so as to apply microtiming from a similar learned phrase to a new example. Because our data does not seem to be composed of phrases in the same way, we did not try this approach. Instead we embodied our intuitions about what structures microtiming into a distance function.

3.3 Our Distance Function

All of the learning algorithms we tried, except for GPR, are based directly on a notion of *distance* between an input note and each note in the training data. Our goal in devising the distance function was to account for the factors that we believe determine notes' microtiming: timbre, position in the rhythmic cycle, and the presence of other notes nearby in time. Our distance function has three components:

- *Timbral distance* is a subjective measure of the degree to which two instruments sound similar and/or have similar rhythmic functions.
- *Metric position distance* is a subjective measure of the degree to which notes at two metric positions have the same rhythmic effect; e.g., the 8th note after beat two is closest to 8th note after beat four.
- *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.

The relative importance of these components was also chosen subjectively. This function is nonnegative and symmetric, but the triangle equality does not necessarily hold, so it is not a true distance metric in the mathematical sense.

4 Machine Learning Algorithms

Our algorithms' input is the MIDI score of a series of quantized drum notes, and the output is an estimate of the microtiming to apply to each note. Since the output is a continuous-valued estimate rather than a discrete-valued decision, we applied *regression* algorithms rather than *classification* algorithms. Standard practice for evaluating the performance of regression algorithms is to calculate the mean squared error between the estimate and the actual value. In our case we used five-fold cross validation.

4.1 K–Nearest–Neighbors / Locally Weighted Linear Regression

K-Nearest-Neighbors (KNN) (Mitchell, 1997) was the simplest algorithm we applied; it outputs a linear combination of the microtimings for the k training notes "nearest" to the given test note. KNN is especially attractive because the running time for estimating the microtiming of m test notes is only $O(mn)$, when there are n training samples, and so KNN would be feasible in live

settings with the training set even being generated on-the-fly. We used the following standard function to compute the similarity $w^{(i,j)}$ given the distance between the notes i and j.

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

τ is called the bandwidth parameter and controls the degree to which distance affects weight. When k equals the size of the training set, K-Nearest-Neighbors is equivalent to Locally Weighted Linear Regression (LWLR) (Mitchell, 1997); this was never the optimal value of k. For KNN, we selected k and τ to minimize the MSE over all our data, resulting in k=26 and τ=24. Note that KNN considers only the distances between each test set element and the training set elements, not distances between training set elements; therefore, a given test note can be close to two test set elements that are far from each other, resulting in a weighted average of microtimings from two distant and thus likely unrelated notes.

4.2 Kernel Ridge Regression

In contrast with KNN, Kernel Ridge Regression (KRR) considers the distances between pairs of elements in the training set when deciding how heavily to weight the influence of relevant training examples (Welling, Accessed 2006). To this end, KRR solves a set of linear equations involving a matrix W consisting of $w^{(i,j)}$ for $1 \le i, j \le n$, which represent the similarity between pairs of notes in the training set. The computational cost of the algorithm depends on the expense of factoring W, but it is quite reasonable. However, for online applications, the training set would need to be prepared beforehand so that W would only need to be factored once.

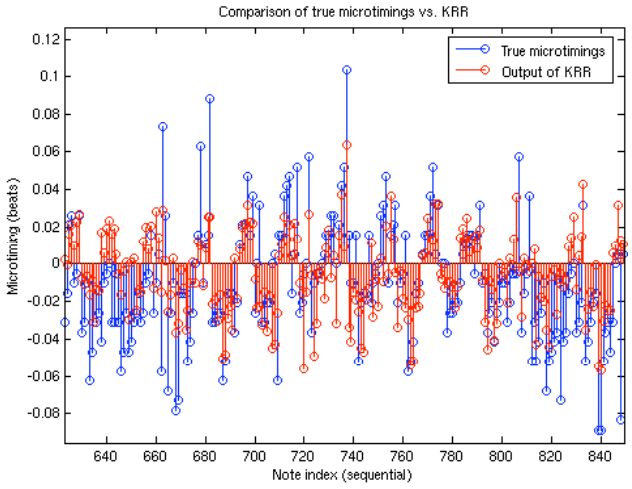

Figure 1: Comparison of the true microtimings versus those predicted by KRR for an excerpt of the *Escola* rhythm.

Figure 1 shows an excerpt of about 11 beats' duration of the results from cross-validation for the rhythm *Escola*. Positive microtiming indicates a delay relative to the beat. Despite the difficult nature of the regression problem, KRR is able to make reasonable estimates of many of the microtimings. For example, the sign of the estimate is very frequently correct. Note how the microtimings tend to oscillate; this could be due to small local tempo fluctuations about the fixed tempo.

KRR sometimes performed slightly better than KNN and sometimes slightly worse (see Fig. 2). Perhaps the suboptimal performance of KRR was related to the fact that we needed to add a small, scaled version of the identity matrix to W in order to make the matrix invertible.

However, we could certainly improve the performance of both KNN and KRR by refining our distance function. On the other hand, searching the space of possible distance functions, or even just of relative weightings of components of a distance function, would be exceedingly time-consuming because we would have to use cross validation to verify the effects of each parameter change, and we would probably not gain as much insight into the problem as from applying other ML algorithms that can automatically fine-tune the distance function.

4.3 Gaussian Processes Regression

The typical formulation of Gaussian Process Regression (GPR) allows automatic fine-tuning[2] of the distance function's weights (Rasmussen, 1996). However, the weights can only be easily optimized for a few particular structures, such as a Euclidean distance. As a result, we needed to reformulate our notion of distance as the weighted Euclidean distance between two feature vectors rather than as an arbitrary function of two input notes. Since our distance function incorporated so many aspects, we required feature vectors containing a few hundred elements. We included features such as the note's onset time within the measure, the note's MIDI velocity, and a representation of the local score surrounding the note, which consisted of the cross product between the various timbres in the sequence and the note locations within a symmetric two beat window. However, given our moderately sized amount of training data (see Table 1), we could estimate on the order of only 10 to 20 weights robustly, so we reduced the dimensionality of the score information using Principal Components Analysis (PCA). A paper by Ellis and Arroyo outlines a similar method for popular music using what they term "eigenrhythms" and "indirhythms" (Ellis & Arroyo, 2004).

Estimating microtimings of test data with GPR takes roughly as long as with KRR assuming that the weights have already been fine-tuned; however, fine-tuning the weights is computationally expensive: each iteration of the optimization requires $O(n^3)$ time. Consequentially, we required 30 minutes to fine-tune the thirteen weights for each rhythm using Matlab on a 2.2GHz dual-processor workstation.

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

5 Results and Future Work

The performance of all four methods, measured by the MSE on 5-fold cross-validation, is strikingly similar (see Fig. 2) even though GPR relied on a very different distance function. This may indicate that our notion of distance needs to consider other features, e.g., a larger window of musical context, or that we simply need more training data. On the other hand, we believe that our results[3] sound much better than the MSE results would indicate. This is because the microtiming output by each ML algorithm is nonzero (so things don't sound quantized), systematic (because the algorithms are not random), and somewhat different for each bar (unlike groove templates, which can still sound artificial for that reason).

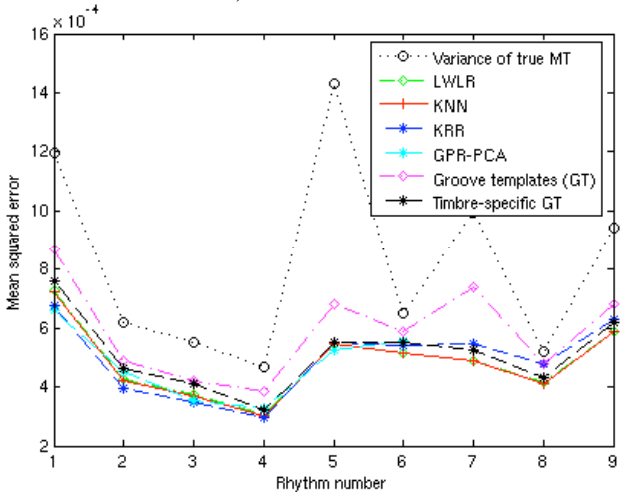

Figure 2: Mean Squared Error of each method. The top line shows the variance of the true microtiming; it can be interpreted as the error from applying no microtiming at all.

We believe that GPR has the most room for improvement in future studies. Further preprocessing of the local score data could be helpful, and there may be better methods such as independent components analysis (ICA) for reducing the dimensionality of the local score that result in eigenvectors more relevant to the regression task. In addition, although fine-tuning the weights for GPR is computationally expensive, it is much faster than using the distance function described in section 3.3 and tuning the weights using cross-validation.

We created several examples using cross-synthesis, where the microtiming deviations from one rhythm are applied to another quantized score. We would like to explore more such creative applications of this work. Finally, although all of our training data comes from Brazilian rhythms, very little of our work embodies specific knowledge of that musical culture, so these methods should be easily adaptable to other forms of music.

[3] All our software and sound examples, including comparisons against groove templates and other techniques, can be found at http://ccrma.stanford.edu/~eberdahl/Projects/Microtiming/

6 Acknowledgments

John Lazarro, Andrew Ng, Yirong Shen, David Wessel.

References

Bilmes J. 1993. Timing is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Timing in Percussive Rhythm, Media Lab. Massachusetts Institute of Technology: Cambridge, Massachusetts

Busse W. 2002. Toward Objective Measurement and Evaluation of Jazz Piano Performance Via MIDI-Based Groove Quantize Templates. Music Perception 19(3), 443-461

Clarke EF. 1999. Rhythm and Timing in Music. In D Deutsch (Ed.), The Psychology of Music, 473-500. Academic Press: San Diego

Clynes M. 1983. Expressive Microstructure in Music, Linked to Living Qualities. In J Sundberg (Ed.), Studies of Music Performance, 76-181. Royal Swedish Academy of Music: Stockholm

Ellis DPW, Arroyo J. 2004. Eigenrhythms: Drum pattern basis sets for classification and generation, Int. Symp. on Music Info. Retr. ISMIR-04: Barcelona (http://www.ee.columbia.edu/~dpwe/pubs/ismir04-eigenrhythm.pdf)

Friberg A, Sundström A. 2002. Swing Ratios and Ensemble Timing in Jazz Performance: Evidence for a Common Rhythmic Pattern. Music Perception 19(3), 333-349

Grachten, J. Ll. Arcos, R. López de Mántaras. 2006. A Case Based Approach to Expressivity-aware Tempo Transformation. Machine Learning Journal. In press

Honing H. 2001. From Time to Time: The Representation of Timing and Tempo. Computer Music Journal 25(3), 50 - 61

Iyer V. 2002. Embodied Mind, Situated Cognition, and Expressive Microtiming in African-American Music Music Perception 19(3), 387–414

Jaffe D. 1986. Ensemble Timing in Computer Music. Computer Music Journal 9(4), 38-48

Mitchell T. 1997. Machine Learning. McGraw Hill

Rasmussen CE. 1996. Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression, Graduate Dept. of Comp. Sci. University of Toronto: Toronto, Canada

Sundberg J, Askenfelt A, Frydén L. 1983. Musical performance. A synthesis-by-rule approach. Computer Music Journal 7, 37–43

Waadeland CH. 2001. It Don't Mean a Thing If It Ain't Got That Swing – Simulating Expressive Timing by Modulated Movements. Journal of New Music Research 30(1), 23-37

Welling M. Accessed 2006. Kernel ridge Regression, (http://www.ics.uci.edu/~welling/classnotes/papers_class/Kernel-Ridge.pdf)

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

Widmer G. 2002. Machine discoveries: A few simple, robust local expression principles. J. New Music Res. 31(1), 37–50