Invited Applications Paper

Music Plus One and Machine Learning

Christopher Raphael CRAPHAEL@INDIANA.EDU
School of Informatics and Computing, Indiana University, Bloomington

Abstract

A system for musical accompaniment is presented in which a computer-driven orchestra follows and learns from a soloist in a concerto-like setting. The system is decomposed into three modules: the first computes a real-time score match using a hidden Markov model; the second generates the output audio by phase-vocoding a preexisting audio recording; the third provides a link between these two, by predicting future timing evolution using a Kalman filter-like model. Several examples are presented showing the system in action in diverse musical settings. Connections with machine learning are highlighted, showing current weaknesses and new possible directions.

1. Musical Accompaniment Systems

Musical accompaniment systems are computer programs that serve as musical partners for live musicians, usually playing a supporting role for music centering around the live player. The types of possible interaction between live player and computer are widely varied. Some approaches create sound by processing the musician's audio, often driven by analysis of the audio content itself, perhaps distorting, echoing, harmonizing, or commenting on the soloist's audio in largely predefined ways, (Lippe, 2002), (Rowe, 1993). Other orientations are directed toward improvisatory music, such as jazz, in which the computer follows the outline of a score, perhaps even composing its own musical part "on the fly" (Dannenberg & Mont-Reynaud, 1987), or evolving as a "call and response" in which the computer and human alternate the lead role (Franklin, 2002), (Pachet, 2004). Our focus here is on a third approach that models the traditional "classical" concerto-type setting in which the computer performs a precomposed musical part in a way that follows a live soloist (Dannenberg & Mukaino, 1988),(Cont et al., 2005),(Raphael, 2009). This categorization is only meant to summarize some past work, while acknowledging that there is considerable room for blending these scenarios, or working entirely outside this realm of possibilities.

The motivation for the *concerto* version of the problem is strikingly evident in the Jacobs School of Music (JSoM) at Indiana University, where most of our recent experiments have been performed. For example, the JSoM contains about 200 student pianists, for whom the concerto literature is central to their daily practice and aspirations. However, in the JSoM the regular orchestras perform only two piano concerti each year using student soloists, thus ensuring that most of these aspiring pianists will never perform as orchestral soloist while at IU. We believe this is truly unfortunate since nearly all of these students have the necessary technical skills and musical depth to greatly benefit from the concerto experience. Our work in musical accompaniment systems strives to bring this rewarding experience to the music students, amateurs, and many others who would like to play as orchestral soloist, though, for whatever reason, don't have the opportunity.

Even within the realm of classical music, there are a number of ways to further subdivide the accompaniment problem, requiring substantially different approaches. The JSoM is home to a large string pedagogy program beginning with students at 5 years of age. Students in this program play solo pieces with piano even in their first year. When accompanying these early-stage musicians, the pianist's role is not simply to *follow* the young soloist, but to *teach* as well, by modeling good rhythm, steady tempo where appropriate, while introducing musical ideas. In a sense, this is the hardest of all classical music accompaniment problems, since the accompanist must be expected to know *more* than the soloist, thus dictating when the accompanist should follow, as well as when and *how* to lead. A coarse approximation to this accompanist

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

role is to provide a rather rigid accompaniment that is not overly responsive to the soloist's interpretation (or errors) — there are several commercial programs that take this approach. The more sophisticated view of the pedagogical music system — one that follows and leads as appropriate — is almost completely untouched, possibly due to the difficulty of modeling the objectives. However, we see this area as fertile for lasting research contributions and hope that we, and others, will be able to contribute to this cause.

An entirely different scenario deals with music that evolves largely without a sense of rhythmic flow, such as in some compositions of Penderecki, Xenakis, Boulez, Cage, and Stockhausen, to name some of the more famous examples. Such music is often notated in terms of seconds, rather than beats or measures, to emphasize the irrelevance of traditional pulse. For works of this type involving soloist and accompaniment, the score can indicate points of synchronicity, or time relations, between various points in the solo and accompaniment parts. If the approach is based solely on audio, a natural strategy is simply to wait until various solo events are detected, and then to *respond* to these events. This is the approach taken by the IRCAM score follower, with some success in a variety of pieces of this type (Cont et al., 2005).

A third scenario, which includes our system, treats works for soloist and accompaniment having a continuing musical pulse, including the overwhelming majority of "common practice" art music. This music is the primary focus of most of our performance-oriented music students at the JSoM, and is the music where our accompaniment system is most at home. Music containing a regular, though not rigid, pulse requires close synchronization between the solo and accompanying parts, as the overall result suffers greatly as this synchrony degrades.

Our system is known interchangeably as the "Informatics Philharmonic," or "Music Plus One" (MPO), due to its alleged improvement on the play-along accompaniment records from *Music Minus One* that inspired our work. For several years we have been collaborating with faculty and students in the JSoM on this traditional concerto setting, in an ongoing effort to improve the performance of our system while exploring variations on this scenario. A video of such a collaboration is contained in `http://www.music.informatics.indiana.edu/papers/icml10`, while also providing further examples relevant to our discussion here. We will present a description of the overall architecture of our system in terms of its three basic components: Listen, Predict, and Play, including several illuminating examples. We also identify open problems or limitations of proposed approaches that are likely to be interesting to the Machine Learning community, and well may benefit from their contributions.

The basic technology required for common practice classical music extends naturally to the *avant garde* domain. In fact, we believe one of the greatest potential contributions of the accompaniment system is in new music composed specifically for human-computer partnerships. The computer offers essentially unlimited virtuosity in terms of playing fast notes and coordinating complicated rhythms. On the other hand, at present, the computer is comparatively weak at providing aesthetically satisfying musical interpretations. Compositions that leverage the technical ability of the accompaniment system, while humanizing the performance through the human soloist's leadership, provide a open-ended musical meeting place for the 21st-century composition and technology. Several compositions of this variety, written specifically for our accompaniment system by Swiss composer and statistician Jan Beran, are presented at the web page referenced above.

2. Overview of Music Plus One

Our system is composed of three sub-tasks called "Listen," "Predict," and "Play." The Listen module interprets the audio input of the live soloist as it accumulates in real-time. In essence, Listen annotates the incoming audio with a "running commentary," identifying note onsets with variable detection latency, using the hidden Markov model discussed in Section 3. A moment's thought here reveals that some detection latency is inevitable since a note must be heard for an instant before it can be identified. For this reason we believe it is hopeless to build a purely "responsive" system — one that waits until a solo note is detected before playing a synchronous accompaniment event: our detection latency is usually in the 30-90 ms. range, enough to prove fatal if the accompaniment is consistently behind by this much. For this reason we model the timing of our accompaniment on the human musician, continually predicting future evolution, while modifying these predictions as more information becomes available. The module of our system that performs this task, Predict, is a Gaussian graphical model quite close to a Kalman Filter, discussed in Section 4. The Play module uses phase-vocoding (Flanagan & Golden, 1966) to construct the orchestral audio output using audio from an accompaniment-only recording. This well-known technique warps the timing of the

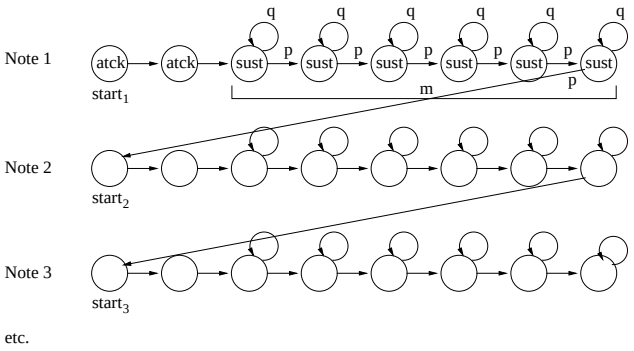

Figure 1. The state graph for the hidden sequence, $x_1, x_2, \ldots$, of our HMM.

original audio without introducing pitch distortions, thus retaining much of the original musical intent including balance, expression, and tone color. The Play process is driven by the output of the Predict module, in essence by following an evolving sequence of future targets like a trail of breadcrumbs.

While the basic methodology of the system relies on old standards from the ML community — HMMs and Gaussian graphical models — the computational challenge of the system should not be underestimated, requiring accurate real-time two-way audio computation in musical scenarios complex enough to be of interest in a sophisticated musical community. The system was implemented for off-the-shelf hardware in C and C++ over a period of about fifteen years by the author, resulting in more than 100,000 lines of code. Both Listen and Play are implemented as separate threads which both make calls to the Predict module when either a solo note is detected (Listen) or an orchestra note is played (Play). What follows is a more detailed look at Listen and Predict.

3. Listen: HMM-Based Score Following

Blind music audio recognition treats the automatic transcription of music audio into symbolic music representations, using no prior knowledge of the music to be recognized. This problem remains completely open, especially with polyphonic (several independent parts) music, where the state of the art remains primitive. While there are many ways one can build reasonable data models quantifying how well a particular audio snippet matches a hypothesized collection of pitches, what seems to be missing is the musical language model. If phonemes and notes are regarded as the atoms of speech and music, there does not seem to be a musical equivalent of the *word*. Furthermore, while music follows simple logic and can be quite predictable, this logic is often cast in terms of higher-level constructs such as meter, harmony and motivic transformation. Computationally tractable models such as note n-grams seem to contribute very little here, while a computationally useful music language model remains uncharted territory.

Our Listen module deals with the much simpler situation in which the music *score* is known, giving the pitches the soloist will play along with their approximate durations. Thus the score following problem is one of *alignment* rather than *recognition*. Score following, otherwise known as on-line alignment, is more difficult than its off-line cousin, since an on-line algorithm cannot consider future audio data in estimating the times of audio events. Thus, one of the principal challenges of on-line alignment is the trade-off between accuracy — reporting the correct times of note events — and latency — the lag in time between the estimated note event time and the time the estimate is made. (Schwarz, 2003) gives a nice annotated bibliography of the many contributions to score following.

3.1. The Listen Model

Our HMM approach views the audio data as a sequence of "frames," $y_1, y_2, \ldots, y_T$, with about 30 frames per second, while modeling these frames as the output of a hidden Markov chain, $x_1, x_2, \ldots, x_T$. The state graph for the Markov chain, described in Figure 1, models the music as a sequence of sub-graphs, one for each solo note, which are arranged so that the process enters the start of the $n+1$st note as it leaves the nth note. From the figure, one can see that each note begins with a short sequence of states meant to capture the *attack* portion of the note. This is followed by another sequence of states with self-loops meant to capture the main body of the note, and to account for the variation in note duration we may observe, as follows.

If we chain together m states which each either move forward, with probability p, or remain in the current state, with probability $q = 1-p$, then the total number of state visits, L, (audio frames) spent in the sequence of m states has a negative binomial distribution

$$P(L = l) = \left(\begin{array}{c} m-1 \\ l-1 \end{array} \right) p^m q^{l-m}$$

for $l = m, m+1, \ldots$. While convenient to represent this distribution with a Markov chain, the asymmetric nature of the negative binomial is also musically reasonable: while it is common for an inter-onset interval (IOI) to be much longer than its nominal length, the reverse is much less common. For each note we choose the note parameters m and p so that $E(T) = m/p$

and $\text{Var}(T) = mq/p^2$ reflect our prior beliefs. If we have seen several performances of the piece in question, we choose m and p according to the method of moments — so that the empirical mean and variance agree with the true mean and variance. Otherwise, we choose the mean according to what the score prescribes, while choosing the variance to capture our lack of knowledge.

In reality, we use a wider variety of note models than depicted in the figure, with variants on the above model for short notes, notes ending with optional rests, notes that are rests, etc, though all following the same essential idea. The result is a network of thousands of states.

Our data model is composed of three features $b_t(y_t), e_t(y_t), s_t(y_t)$ assumed to be conditionally independent given the state

$$P(b_t, e_t, s_t|x_t) = P(b_t|x_t)P(e_t|x_t)P(s_t|x_t).$$

The first feature, b_t, measures the local "burstiness" of the signal, particularly useful in distinguishing between note attacks and steady state behavior — observe that we distinguished between the attack portion of a note and and steady state portion in Figure 1. The 2nd feature, e_t measures the local energy, useful in distinguishing between rests and notes. By far, however, the vector-valued feature s_t is the most important, as it is well-suited to making pitch discriminations, as follows.

We let f_n denote the frequency associated with the nominal pitch of the nth score note. As with any quasi-periodic signal with frequency f_n, we expect that the audio data from the nth note will have a magnitude spectrum composed of "peaks" at integral multiples of f_n. This is modeled by the Gaussian mixture model depicted in Figure 2

$$p_n(j) = \sum_{h=1}^{H} w_h N(j; \mu = hf_n, \sigma^2 = (\rho h f_n)^2)$$

where $\sum_h w_h = 1$ and $N(j; \mu, \sigma^2)$ is a discrete approximation of a Gaussian distribution. We define s_t to be the magnitude spectrum of y_t, normalized to sum to constant value, C. If we believe the nth note is sounding in the tth frame, we regard s_t as the histogram of a random sample of size C. Thus our data model becomes the multinomial distribution

$$P(s_t|x_t \in \text{ note } n) = \frac{C!}{\prod_j s_t(j)!} \prod_j p_n(j)^{s_t(j)} \qquad (1)$$

This model describes the part of the audio spectrum due to the soloist reasonably well. However, our actual signal will receive not only this solo contribution,

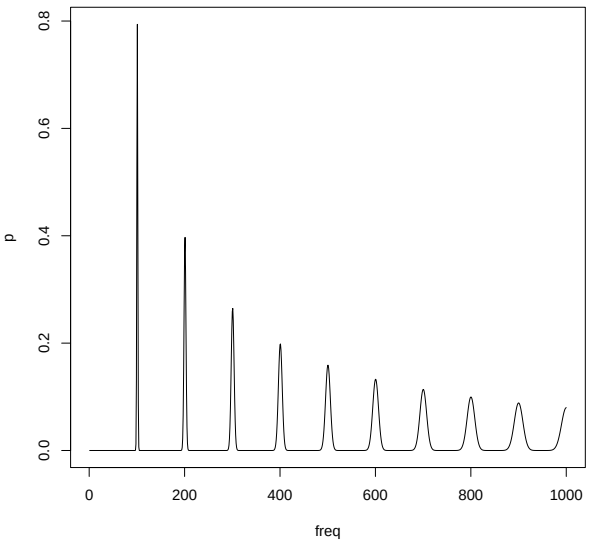

Figure 2. An idealized note spectrum modeled as a mixture of Gaussians.

but audio *generated* by our accompaniment system as well. If the accompaniment audio contains frequency content that is confused with the solo audio, the result is the highly undesirable possibility of the accompaniment system *following itself* — in essence, chasing its own shadow. To a certain degree, the likelihood of this outcome can be diminished by "turning off" the score follower when the soloist is not playing; of course we do this. However, there is still significant potential for shadow-chasing since the pitch content of the solo and accompaniment parts is often similar.

Our solution to this difficulty is to directly model the accompaniment contribution to the audio signal we receive. Since we *know* what the orchestra is playing (our system generates this audio), we add this contribution to the data model. More explicitly, if q_t is the magnitude spectrum of the orchestra's contribution in frame t, we model the conditional distribution of s_t using Eqn. 1, but with $p_{k,n} = \lambda p_n + (1 - \lambda)q_t$ for $0 < \lambda < 1$ instead of p_n.

This addition creates significantly better results in many situations. The surprising difficulty in actually implementing the approach, however, is that there seems to be only weak agreement between the *known* audio that our system plays through the speakers and the accompaniment audio that *comes back* through the microphone. Still, with various averaging tricks in the estimation of q_t, we can nearly eliminate the undesirable shadow-chasing behavior.

3.2. On-Line Interpretation of Audio

Perhaps one of the main virtues of the HMM-based score follower is the grounding it gives to navigating the accuracy-latency trade-off. One of the worst things a score follower can do is report events before they have occurred. In addition to the sheer impossibility of producing accurate estimates in this case, the musical result often involves the accompanist arriving at a point of coincidence before the soloist does. When the accompanist "steps on" the soloist in this manner, the soloist must struggle to regain control of the performance, perhaps feeling desperate and irrelevant in the process. Since the consequences of false positives are so great, the score follower must be reasonably certain that a note event has already occurred before reporting its location. We handle this as follows.

Every time we process a new frame of audio we recompute the "forward" probabilities, $p(x_t|y_1,\ldots,y_t)$, for our current frame, t. Listen waits to detect note n until we are sufficiently confident that its onset is in the past. That is, until

$$P(x_t \geq \mathrm{start_n}|\mathrm{y_1},\ldots,\mathrm{y_t}) \geq \tau$$

In this expression, start_n represents the initial state of the nth note model, as indicated in Figure 1, which is either before, or after all other states in the model. Suppose that t^* is the first frame where the above inequality holds. When this occurs, our knowledge of the note onset time can be summarized by the function of t:

$$P(x_t = \mathrm{start}_n|y_1,\ldots,y_{t^*})$$

which we compute using the forward-backward algorithm. Occasionally this distribution conveys uncertainty about the onset time of the note, say, for instance, if it has high variance or is bimodal. In such a case we simply do not report the onset time of the particular note, believing it is better to remain silent than provide bad information. Otherwise, we estimate the onset as

$$\hat{t}_n = \arg\max_{t \leq t^*} P(x_t = \mathrm{start}_n|y_1,\ldots,y_{t^*}) \qquad (2)$$

and deliver this information to the Predict module.

Several videos demonstrating the ability of our score following can be seen at the aforementioned web site. One of these simply plays the audio while highlighting the locations of note onset detections at the times they are made, thus demonstrating detection latency — what one sees lags slightly behind what one hears. A second video shows a rather eccentric performer who ornaments wildly, makes extreme tempo changes, plays wrong notes, and even repeats a measure, thus demonstrating the robustness of the the score follower.

4. Predict: Modeling Musical Timing

As discussed in Section 2, we believe a purely *responsive* accompaniment system can not achieve acceptable coordination of parts in the range of common practice "classical" music we treat, thus we choose to schedule our accompaniment through prediction rather than response. Our approach is based on a model for musical timing. In developing this model, we begin with three important traits we believe such a model must have.

1. Since our accompaniment must be constructed in real time, the computational demand of our model must be feasible in real time.

2. Our system must improve with rehearsal. Thus our model must be able to automatically train its parameters to embody the timing nuances demonstrated by the live player in past examples. This way our system can better *anticipate* the future musical evolution of the current performance.

3. If our rehearsals are to be successful in guiding the system toward the desired musical end, the system must "sightread" (perform without rehearsal) reasonably well. Otherwise, the player will become distracted by the poor ensemble and not be able to demonstrate what she wants to hear. Thus there must be a neutral setting of parameters that allows the system to perform reasonably well "out of the box."

4.1. The Timing Model

We first consider a timing model for a single musical part. Our model is expressed in terms of two hidden sequences, $\{t_n\}$ and $\{s_n\}$ where t_n is the time, in seconds, of nth note onset and s_n is the tempo, in seconds per beat, for the nth note. These sequences evolve according to

$$s_{n+1} = s_n + \sigma_n \qquad (3)$$
$$t_{n+1} = t_n + l_n s_n + \tau_n \qquad (4)$$

where l_n is the length of the nth event, in beats.

With the "update" variables, $\{\sigma_n\}$ and $\{\tau_n\}$, set to 0, this model gives a literal and robotic musical performance with each inter-onset-interval, $t_{n+1} - t_n$, consuming an amount of time proportional to its length in beats, l_n. The introduction of the update variables allow time-varying tempo through the $\{\sigma_n\}$, and elongation or compression of note lengths with the $\{\tau_n\}$. We further assume that the $\{(\sigma_n, \tau_n)^t\}$ are independent with $(\sigma_n, \tau_n)^t \sim N(\mu_n, \Gamma_n)$ and $(s_1, t_1)^t \sim N(\mu_0, \Gamma_0)$, thus leading to a joint Gaussian model on all model

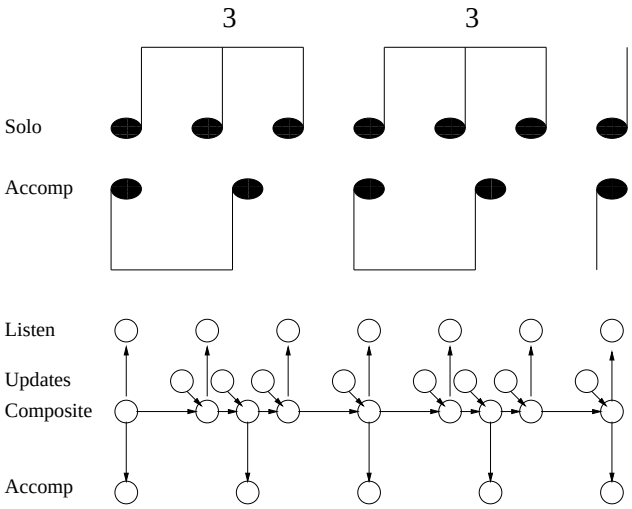

Figure 3. **Top:** Two musical parts generate a *composite* rhythm when superimposed. **Bot:** The resulting graphical model arising from the composite rhythm.

variables. The rhythmic interpretation embodied by the model is expressed in terms of the $\{\mu_n, \Gamma_n\}$ parameters. In this regard, the $\{\mu_n\}$ vectors represent the *tendencies* of the performance — where the player tends to speed up ($\sigma_n < 0$), slow down ($\sigma_n > 0$), and stretch ($\tau_n > 0$), while the $\{\Gamma_n\}$ matrices capture the repeatability of these tendencies.

It is simplest to think of Eqns. 3-4 as a timing model for single musical part. However, it just as reasonable to view these Eqns. as a timing model for the *composite* rhythm of the solo *and* orchestra. That is, consider the situation, depicted in Figure 3, in which the solo, orchestra, and composite rhythms have the following musical times (in beats):

solo	0	1/3		2/3	1	4/3		5/3	2
accomp	0		1/2		1		3/2		2
comp.	0	1/3	1/2	2/3	2	4/3	3/2	5/3	2

The $\{l_n\}$ for the composite would be found by simply taking the differences of rational numbers forming the composite rhythm: $l_1 = 1/3, l_2 = 1/6$, etc. In what follows we regard Eqns. 3-4 as a model for this composite rhythm.

The *observable* variables in this model are the solo note onset estimates produced by Listen and the *known* note onsets of the orchestra (our system constructs these during the performance). Suppose that n indexes the events in the composite rhythm having associated solo notes, estimated by the $\{\hat{t}_n\}$. Additionally, suppose that n' indexes the events having associated orchestra notes with onset times, $\{o_{n'}\}$. We model

$$\hat{t}_n \quad = \quad t_n + \epsilon_n$$

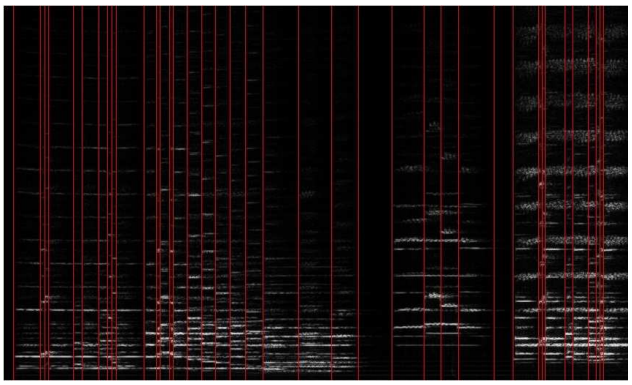

Figure 4. A "spectrogram" of the opening of the 1st movement of the Dvoŕák Cello concerto. The horizontal axis of the figure represents time while the vertical axis represents frequency. The vertical lines show the note times for the orchestra.

$$o_{n'} \quad = \quad t_{n'} + \delta_{n'}$$

where $\epsilon_n \sim N(0, \rho_s^2)$ and $\delta_{n'} \sim N(0, \rho_o^2)$. The result is the Gaussian graphical model depicted in the bottom panel of Figure 3. In this figure the row labeled "Composite" corresponds to the $\{(s_n, t_n)\}$ variables of Eqns. 3-4, while the row labeled "Updates" corresponds to the $\{(\sigma_n, \tau_n)\}$ variables. The "Listen" row is the collection of estimated solo note onset times, $\{\hat{t}_n\}$ while the "Accompaniment" row corresponds to the orchestra times, $\{o_{n'}\}$.

4.2. The Model in Action

With the model in place we are ready for real-time accompaniment. In our first rehearsal we initialize the model so that $\mu_n = 0$ for all n. This assumption does not preclude the possibility of the model correctly interpreting and following tempo changes or other rhythmic nuances of the soloist; rather, it states that we *expect* the timing to evolve strictly according to the tempo when looking forward.

In real-time accompaniment, our system is concerned only with scheduling the currently pending orchestra note time, $o_{n'}$. The time of this note is initially scheduled when we play the previous orchestra note, $o_{n'-1}$. At this point we compute the new mean of $o_{n'}$, conditioning $o_{n'-1}$ and whatever other variables have been observed, and schedule $o_{n'}$ accordingly. While we wait for the currently-scheduled time to occur, the Listen module may detect various solo events, $\hat{t}_n$. When this happens we *recompute* the mean of $o_{n'}$, conditioning on this new information. Sooner or later the actual clock time will catch up to the currently-scheduled time, at which point the orchestra note is played. Thus an or-

chestra note may be rescheduled many times before it is actually played. A particularly instructive example involves a run of many solo note culminating in a point of coincidence with the orchestra. As each solo note is detected we refine our estimate of the desired point of coincidence, thus gradually "honing in" on the this point of arrival. It is worth noting that very little harm is done when Listen fails to detect a solo note. We simply predict the pending orchestra note conditioning on the variables we *have* observed.

The web page given before contains a video that demonstrates this process. The video shows the estimated solo times from our score follower appearing as green marks on a spectrogram. Predictions of our accompaniment system are shown as analogous red marks. One can see the pending accompaniment event "jiggling" as new solo notes are estimated, until finally the currently-predicted time passes.

The role of Predict is to "schedule" accompaniment notes, but what does this really mean in practice? Recall that our program plays audio by phase-vocoding (time-stretching) an orchestra-only recording. A time-frequency representation of such an audio file for the 1st movement of the Dvoŕák Cello concerto is shown in Figure 4. In preparing this audio for our accompaniment system, we perform an off-line score alignment to determine where the various orchestra notes occur, as marked with vertical lines in the figure. Scheduling a note simply means that we change the phase-vocoder's play rate so that it arrives at the appropriate audio file position (vertical line) at the scheduled time. Thus the play rate is continually modified as the performance evolves.

After one or more "rehearsals," we *adapt* our timing model to the soloist to better anticipate future performances. To do this we first perform an off-line estimate of the solo note times using Eqn. 2, only conditioning on the entire sequence of frames, $y_1, \ldots, y_T$, using the forward-backward algorithm. Using one or more such rehearsals we can iteratively reestimate the model parameters $\{\mu_n\}$ using the EM algorithm, resulting in both measurable and perceivable improvment of prediction accuracy. While, in principle, we can also estimate the $\{\Gamma_n\}$ parameters, we have observed little or no benefit from doing so.

In practice we have found the soloist's interpretation to be a bit of a "moving target." At first this is because the soloist tends to compromise somewhat in the initial rehearsals, pulling the orchestra in the desired *direction*, while not actually reaching the target interpretation. But even after the soloist seems to settle down to a particular interpretation on a given day, we often observe further "drift" of the interpretation over subsequent meetings, due to the changeable nature of musical ideas. For this reason we train the model using the most recent several rehearsals, thus facilitating the continually evolving nature of musical interpretation.

5. Musical Expression and Machine Learning

Our system learns its musicality through "osmosis." If the soloist plays in a musical way and the orchestra manages to closely follow the soloist, then we hope the orchestra will *inherit* this musicality. This manner of learning by imitation works well in the concerto setting, as the division of authority between the players is rather extreme, mostly granting the "right of way" to to the soloist. However, many other musical scenarios do not so clearly divide the roles into *leader* and *follower*. Our system is least at home in this middle ground.

Our timing model of Eqns. 3-4 is over-parametrized, with more degrees of freedom than there are notes. We make this modeling choice because it is hard know exactly which degrees of freedom are needed ahead of time, so we use the training data from the soloist to help sort this out. Unnecessary learned parameters may contribute some noise to the resulting timing model, but the overall result is acceptable.

While this approach works reasonably well in the concerto setting, it is less reasonable when our system needs a sense of musicality that acts independently, or perhaps even in opposition, to what other players do. Such a situation occurs with the early-stage accompaniment problem discussed in Section 1, as here one cannot learn the desired musicality from the live player. Perhaps the accompaniment *antithesis* of the concerto setting is the opera orchestra, in which the "accompanying" ensemble is often on equal footing with the soloists. We observed the nadir of our system's performance in an opera rehearsal where our system served as rehearsal pianist. What these two situations have in common is that they require an accompanist with independent musical knowledge and goals.

One possible line of improvement is simply decreasing the model's freedom — surely the player does not wish to change the tempo *and* apply tempo-independent note length variation on every note. One possibility adds a hidden discrete process that "chooses," for each note, between three possibilities: no variation of either kind, or variation of *either* tempo *or* note length. Of these, the choice of neither variation would be the most likely, thus biasing the model toward simpler musical

interpretations, adopting the point of view of "Ockham's razor" with regard to interpretation. While exact inference is no longer possible with such a model, we expect that one can make approximations that will be good enough to realize the full potential of the model, whatever that is.

Perhaps a more interesting approach analyzes the musical score to choose the locations requiring additional degrees of freedom. One can think of this approach as adding "joints" to the musical structure so that it deforms into musically-reasonable shapes as a musician applies external force. Here there is an interesting connection with the work on expressive synthesis, such as (Widmer & Goebl, 2004), in which one algorithmically constructs an expressive rendition of a previously-unseen piece of music, using ideas of machine learning. One idea here is to associate various score situations, defined in terms of local configurations of measurable score features, with interpretive actions. The associated interpretive actions are learned by estimating timing and loudness parameters from a performance corpus, over all "equivalent" score locations. Such approaches are far more ambitious than our present approach to musicality, as they try to understand expression *in general*, rather than in a specific musical context.

The understanding and synthesis of musical expression is surely one of the grand challenges facing music-science researchers, and while progress has been achieved in recent years, it would still be fair to call the problem "open." One of the principal challenges here is that one cannot directly map observable surface-level attributes of the music, such as pitch contour or local rhythm context, into interpretive actions, such as delay, or tempo or loudness change. Rather, there is a murky intermediate level in which the musician comes to some understanding of the musical *meaning*, and on which the interpretive decisions are conditioned. This meaning comes from different aspects of the music. Some comes from structural aspects, for example, as the way the end of a section or phrase is reflected with a sense of closure. Some meaning comes from prosodic aspects, analogous to speech, such as anticipation and points of arrival. A third aspect of meaning describes an overall character or affect of a section of music, such as excited or calm. While there is no official taxonomy of musical interpretation, most discussions on this subject stem from such intermediate identifications and the interpretive actions they require.

From the machine learning point of view it is impossible to learn anything useful from a single example, thus one must group together many examples of the same musical situation in order to learn their associated interpretive actions. Thus it seems natural to model the music in terms of some latent variables that implicitly categorize individual notes or sections of music. What should the latent variables be, and how can one describe the dependency structure among them? While we cannot answer these questions, we see in them a good deal of depth and challenge, and recommend this problem to the musically inclined members of the ML community with great enthusiasm.

References

Cont, A., , Schwarz, D., and Schnell, N. From boulez to ballads: training ircam's score follower. In *Proc. of the International Computer Music Conference*, pp. 241–248, 2005.

Dannenberg, R. and Mukaino, H. New techniques for enhanced quality of computer accompaniment. In *Proc. of the 1988 International Computer Music Conference*, pp. 243–249, 1988.

Dannenberg, Roger and Mont-Reynaud, Bernard. Following an improvisation in real time. In *Proc. of the 1987 International Computer Music Conference*, pp. 241–248, 1987.

Flanagan, J. L. and Golden, R. M. Phase vocoder. *Bell Systems Technical Journal*, 45(Nov):1493–1509, 1966.

Franklin, Judy. Improvisation and learning. In *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.

Lippe, C. Real-time interaction among composers, performers, and computer systems. *Information Processing Society of Japan, SIG Notes*, 123:1–6, 2002.

Pachet, Francois. Beyond the cybernetic jam fantasy: The continuator. *IEEE Computer Graphics and Applications*, 24(1):31–35, 2004.

Raphael, C. Current directions with music plus one. In *Proc. of the 6th Sound and Music Computing Conference*, pp. 71–76, 2009.

Rowe, Robert. *Interactive Music Systems*. MIT Press, 1993.

Schwarz, D. Score following commented bibliography, 2003. URL `http://www.ircam.fr/equipes/temps-reel/suivi/bibliography.html`.

Widmer, G. and Goebl, W. Computational models for expressive music performance: The state of the art. *Journal of New Music Research*, 2004.