Mach Learn (2006) 65:361–387
DOI 10.1007/s10994-006-8751-3

Modeling, analyzing, and synthesizing expressive piano performance with graphical models

Graham Grindlay · David Helmbold

Received: 23 September 2005 / Revised: 17 February 2006 / Accepted: 27 March 2006 / Published online: 21 June 2006
Springer Science + Business Media, LLC 2006

Abstract Trained musicians intuitively produce expressive variations that add to their audience's enjoyment. However, there is little quantitative information about the kinds of strategies used in different musical contexts. Since the literal synthesis of notes from a score is bland and unappealing, there is an opportunity for learning systems that can automatically produce compelling expressive variations. The ESP (Expressive Synthetic Performance) system generates expressive renditions using hierarchical hidden Markov models trained on the stylistic variations employed by human performers. Furthermore, the generative models learned by the ESP system provide insight into a number of musicological issues related to expressive performance.

Keywords Graphical models · Hierarchical hidden Markov models · Music performance · Musical information retrieval

1. Introduction

Skilled human performers interpret the pieces they play by introducing expressive variations to the performed notes, and these variations add to the enjoyment of most[1] human listeners. Although there are many ways in which a performer can introduce variation, perhaps two of the most important are changes in tempo and dynamics. Tempo change or *rubato* is a non-linear warping of the regular grid of beats that defines time in a score. Variations in

Editor : Gerhard Widmer

G. Grindlay (✉)
Media Laboratory, Massachusetts Institute of Technology
e-mail: grindlay@media.mit.edu

D. Helmbold
Computer Science Deptartment, University of California, Santa Cruz

[1] Of course each individual will have their own preferences, but it is reasonable to assume that the top concert pianists are popular because their performances include something more than a precise rendition of the composer's score.

the loudness or *dynamics* (referred to as *velocity* in the case of piano) of individual notes is another commonly employed performance strategy. Other ways to add expression to a piano piece include *articulation* (the gap or overlap between adjacent notes) and the use of the pedals. Our work focuses on tempo variations and considers dynamics to a lesser extent.

We present a general framework for capturing the statistical relationship between compositional score structure and performance variation. Our system uses a dynamic Bayesian network (DBN) representation of a hierarchical hidden Markov model (HHMM) to learn these relationships automatically from a corpus of training data. Once trained, the model can be used to generate expressive performances (tempo curves) of new pieces not in the training set. Furthermore, it provides a statistical generative model of how performers deviate from the written score that can be used for a variety of purposes.

Although generative models like HHMMs are standard machine learning techniques, they have not, to our knowledge, been used to model expressive performances. Our goal is to use as little musicological insight as possible, allowing the latent states in the model to learn the important musical contexts and their appropriate performance variations. One of the main contributions of this work is to illustrate the flexibility and power of this approach when applied to a variety of musicological questions. In particular, we use the learned models to help answer the following musicological questions.

– How consistent are the performances of human individuals?
– How do the performances of professional and trained non-professional pianists compare?
– Can an automated system recognize an individual performer's stylistic variations and/or can the system quantitatively identify those performers with qualitatively distinctive styles (to human listeners)?
– Is there some unifying structure to the variations in performances by top professionals? What about the performances of amateurs?

These questions are important as their answers will increase our understanding of the processes and strategies used by talented musicians to produce memorable performances. Additionally, answers to these questions may enable a number of music information retrieval applications, such as performer search and classification.

Most people would judge the literal rendition of a musical score to be significantly less interesting than a performance of that piece by even a moderately skilled musician. This makes methods for "bringing life" to musical scores useful for several reasons. Such a system could be a boon to composers, who may find the ability to hear their music as it is likely to be performed extremely helpful during the composition process. One might also imagine a system capable of providing tireless accompaniment to a soloist or even a teaching system that could evaluate and critique student performances. Additionally, we believe that by examining the structure of a trained model, it may be possible to gain musicological insight into the complex relationships between compositional structure and expressive performance.

Our current work focuses on modeling tempo variation in piano melodies. In addition to being a very popular instrument, the piano is attractive because of the inherent constraints on note variation. A piano performer can only vary the timing and loudness of notes. In contrast, most other popular instruments have additional hard-to-measure ways of varying the performance of each note (such as vibrato, glissando, and the ability to vary the dynamics throughout the note). Our techniques can be easily generalized to other instruments given a suitable encoding of the performance variations. The present emphasis on tempo variation is entirely due to limitations imposed by the majority of our data. Results by other researchers indicate that tempo is harder to learn than dynamics (Tobudic & Widmer, 2003). We also present positive results for combined tempo and dynamics synthesis using a smaller (but richer) performance data set (see Section 8).

Statistical musicology is still in its infancy and it is extremely difficult to collect large amounts of quantitative data. We are deeply indebted to Bruno Repp for making two data sets available to us. One data set contains tempo information (only) from 28 performances of Robert Schumann's romantic piano piece Träumerei, Op. 15 No. 7 by 24 world-class professional pianists (there are three performances each by Cortot and Horowitz). This information was extracted by hand from commercially distributed CDs. The other data set consists of performances of Träumerei, Chopin's Prelude No. 15 in D♭, and Chopin's Etude Op. 10 No. 3. by 10 graduate students in the Yale School of Music recorded in MIDI format.

We employ similar learning strategies for the two data sets using features from both the score and the performances to build the models. Each model state represents a "musical context" and the transitions between states represent the changing context throughout a piece. In our two-level hierarchical models the top-level abstract states represent context on the level of a musical phrase.[2] The production states are children of a top-level abstract state and represent the various note-level contexts found within their parent's phrasal context. The training data contains a sequence of notes from the score and how they are performed. Each note i in the score has its own vector of score features denoted f_i^s, and an associated vector of performance features, f_i^p, describing how the note was actually played. Each production state contains a joint distribution $P(f^s, f^p)$. We represent performances as the sequence of performance feature vectors, $\boldsymbol{f^p}$, for the played notes. Similarly, a score can be represented as a sequence of score features, $\boldsymbol{f^s}$. Given a set of performances and their corresponding scores, the training process attempts to find the (hierarchical) HMM parameters, $\boldsymbol{\theta}$, that maximize the likelihood of the given feature sequences. Once the model is trained, a performance of a new score, $\boldsymbol{f^{s\prime}}$, can be synthesized by finding the sequence of performance features (approximately) maximizing $P(\boldsymbol{f^p} \mid \boldsymbol{f^{s\prime}}, \boldsymbol{\theta})$. Given an additional performance, $\boldsymbol{f^{p\prime}}$, of a (possibly new) score, $\boldsymbol{f^{s\prime}}$, $P(\boldsymbol{f^{p\prime}} \mid \boldsymbol{f^{s\prime}}, \boldsymbol{\theta})$ gives a relative indication of how closely the new performance fits the trained model.

Because our goals include understanding how skilled performers deviate from the written score, we are interested in models that are interpretable. To this end, we train using an entropic bias along with pruning techniques (Brand, 1999b) to encourage sparseness in the learned models (i.e. relatively low state interconnectivity). This results in models that are often superior in terms of data fit and generalization than what classical training methods (EM, gradient descent, etc.) might produce. Additionally, because of their sparsity, these models are likely to be more interpretable than their classical counterparts.

We conducted a variety of experiments to test the usefulness of the ESP system as a general modeling framework for the synthesis and analysis of expressive performance (see Section 7). Despite a difficult learning task, our results were generally quite good. The results of an experiment in which we synthesized tempo curves using trained models show that ESP is capable of generating performances of previously unseen scores that follow many of the general trends of human performance. We also conducted several experiments designed to test the system's use in performer comparison tasks. A comparison of the performance models of 24 famous performers and 10 student performers seem to confirm earlier observations (Repp, 1997) about the differences in stylistic variation between professional and student

[2] A musical phrase is a relatively self-contained section of music somewhat analogous to a sentence in language. We consider the low-level phrasing information often available in commercial scores. For the pieces in our dataset, each phrase typically contains between 3 and 15 notes.

performers. Additionally, we investigated the ability of the ESP system to recognize individual performers. We conducted a series of three-fold cross validated experiments in which a model was trained on two performances by a specific performer and then used to recognize a third performance by that performer. When trained on two performances by Alfred Cortot, the system was successfully able to pick out a third Cortot performance from amongst a set of 36 performances by other performers. A similar experiment conducted using performances by Vladimir Horowitz was somewhat less successful as the test performance was recognized in one of three trials. This experiment was also carried out for two student performers with good results.

The remainder of the paper is organized as follows. We begin in Section 2 by reviewing some of the relevant machine learning and computational musicology work. Then in Section 3 we discuss the data sets and features used in our experiments. In Section 4 we review the hierarchical hidden Markov model and present the model used in the ESP system. Section 5 describes the training algorithms in the ESP system and Section 6 describes how ESP synthesizes performances from novel scores. Section 7 discusses tempo-based experiments using the ESP system. In Section 8 we describe an earlier version of the ESP system that uses a different modeling scheme handling both tempo and dynamics. Finally, we summarize and discuss our findings in Section 9.

2. Related work

Artificial intelligence and machine learning techniques have found a variety of applications to musical problems in recent years. Much of this work revolves around the so-called "transcription problem" (trying to extract the symbolic representation of a piece of music from raw audio) (Raphael, 2002; Cemgil, Kappen and Barber, 2003; Scheirer, 1995; Dixon, 2001) and the related problems of tempo tracking and rhythm quantization (Lang & de Freitas, 2005; Cemgil and Kappen, 2003). The problems of modeling aspects of composition and performance style have received somewhat less attention. Since two recent overviews (Widmer & Goebl, 2004; de Mántaras & Arcos, 2002) contain extensive references, we mention only the most recent and closely related work here.

Wang et al. use HMMs to learn and synthesize a conductor's movements (Wang et al., 2003). Their work uses a single HMM where each state models a joint distribution over the music (energy, beat, and pitch features) and the positions and velocities of markers on the joints of a conductor. Wang et al. build on earlier work by Brand using HMMs to synthesize lip movements matching speech (Brand, 1999a) and by Brand and Hertzmann to model different styles of bipedal movement (Brand & Hertzmann, 2000). We adapt the ideas in the above papers to the music performance domain and extend the models to hierarchical HMMs using Murphy and Paskin's dynamic Bayesian network based algorithms (Murphy & Paskin, 2002).

Whereas the above approaches use one HMM with dual outputs, Casey creates a fusion of two different drum styles using a pair of HMMs (Casey, 2003). Each HMM's topology captures the high-level structure of a particular drumming style while the emission distributions capture the low-level audio produced by the style. Casey finds a correspondence between the states in the two HMMs and uses this correspondence to combine the low level audio of one style with the high level structure of the other.

Raphael uses Bayesian networks in his *Music Plus One* system for providing performer-specific real time accompaniment (Raphael, 2002b). The structure of the Bayesian network is constructed from the score and then trained with a series of "rehearsals". After training, the system provides expressive real-time accompaniment specialized for the piece on which it was trained.

Work very close in spirit to our own, but using different techniques, is that of Widmer and Tobudic. They model and synthesize expressive performances of piano melodies using a system that automatically learns rules from musical data (Widmer, 2003). This system extracts sets of simple rules, each of which partially explains an aspect of the relationship between score and performance. These simple rules are then combined into larger rules via clustering, generalization, and heuristic techniques. Widmer and Tobudic describe a multi-scale representation of piano tempo and dynamics curves which they use to learn expressive performance strategies (Widmer & Tobudic, 2003). This approach abstracts score deviation temporally and learns how these deviations are applied at both the note and phrase levels via rule-induction algorithms. In more recent work they use nearest neighbor techniques to predict how phrases are played (Tobudic & Widmer, 2003; Tobudic & Widmer, 2005). Our work differs from that of Tobudic and Widmer in that we employ generative statistical models (hierarchical HMMs) to learn the performer's deviations from the score rather than rule-based or instance-based techniques. The hierarchical models capture the phrasal context within a piece as well as the individual note contexts within a phrase.

Archos and López de Mántaras use non-parametric case-based reasoning in their *SaxEx* system (Arcos & de Mántaras, 2001) which renders expressive saxophone melodies from examples. Their system utilizes a database of transformations from scores to expressive performances with features such as dynamics, rubato, vibrato, and articulation. *SaxEx* synthesizes new performances by looking in its database for similar score features and then applying the corresponding expressive transformations. Conflicts that arise when matching score features to entries in the database are resolved with heuristic techniques.

Bresin, Friberg, and Sundberg use a rule-based approach in *Director Musices*, a system for creating expressive performances from raw scores (Bresin, Friberg and Sundberg, 2002). This system is based on a set of hand-crafted rules for synthesizing piano performances. These rules are parametric, providing a means for tuning the contribution of each rule in the synthesis of an expressive performance.

Although much of the research that applies machine learning to expressive performance has focused almost exclusively on synthesis, Stamatatos and Widmer describe a system for classifying performances (Stamatatos & Widmer, 2005). This system maintains a database of professional performers and associates each with certain types of performance features. Then, given a performance, the system uses an ensemble of simple classification algorithms to identify the most likely performer.

Saunders et al. (2004) also describe a system for performer recognition. They use string kernels to represent performances as sequences of prototypical tempo-loudness "shapes". These shapes represent changes in tempo and loudness over a short time period (roughly two beats). Once encoded, they use machine learning techniques such as kernel partial least squares and support vector machines, to recognize the performances of famous pianists.

3. Data

Statistical musicology is in its infancy, and it is still extremely difficult to collect large amounts of quantitative data. We are deeply indebted to Bruno Repp for making two data sets available to us. The first set consists of performances of Schumann's Träumerei, Chopin's

Table 1 Data set summary. This table shows the number of performers, the number of usable performances, the length of the performances in notes, and whether or not dynamics (volume) information is available for each piece used in our experiments

	Professional	Student		
Piece	Träumerei	Träumerei	Prelude	Etude
Performers	24	10	7	7
Performances	28	10	16	7
Dynamics	no	yes	yes	yes
Length (notes)	117	117	131	27

Prelude No. 15 in D♭, and the first six bars of Chopin's Etude Op. 10 No. 3. by 10 graduate students in the Yale School of Music. The performances are in MIDI format (containing both tempo and velocity information) and were recorded on a Yamaha Disklavier[3] piano (see (Repp, 1992) for more details). For Träumerei and Etude Op. 10 No. 3. there is one performance per performer. For Chopin's Prelude No. 15, each performer recorded three takes of each piece with the exception of performer EK who only recorded two takes per song. The performances of all three pieces were matched to their corresponding scores using a simple dynamic programming-based string matching algorithm. We discarded several performances because of their poor match to the scores, and call the remaining performances the *student data set*.

Our second data set, which we refer to as the *professional data set*, contains timing information (only) for 28 performances of Träumerei by 24 famous pianists (there are three performances by both Cortot and Horowitz). This information was extracted by hand from commercially distributed CDs, and contains the duration of inter-onset intervals in milliseconds. The *inter-onset interval* for a note is the time between the initiation of the note and the initiation of the subsequent note. Since notes often overlap, inter onset intervals are not the same as note duration.

The inter-onset intervals are defined relative to a half-beat grid (eighth notes), and longer notes may have their inter-onset intervals evenly split into several half-beat segments. Since each inter-onset value corresponds to the same scored duration, the minutes per beat for a note is a simple re-scaling of the note's first inter-onset interval in the data. Inverting this gives us the note's tempo in terms of beats per minute.

There is a problem with the last note as it has no following note and thus no defined inter-onset-interval. Perhaps the best thing would be to use the last note's duration, but that value is not available in our professional data set. Therefore we opted to simply duplicate the preceeding note's tempo value. Furthermore, we focus on the melody line, dropping other notes (including grace notes) from consideration.

For each remaining note in a performance, we normalize its tempo (beats-per-minute) by dividing by the beats-per-minute for the entire performance. This gives the relative speed of performance at each note. The *tempo factor* feature is the logarithm of the relative speeds. We take the logarithm so that slower-than-average and faster-than-average values are represented on the same numeric scale.[4] We also include the first-order differences of the tempo factors (i.e. how much the tempo changes from one note to the next note). This is not only useful for

[3] The Disklavier is a real piano outfitted with digital sensors that enable it to record performances directly to MIDI.

[4] After taking the logarithm, a note played at the average tempo has a tempo factor of 0, one played twice as fast has a tempo factor of 1, and one played half as fast has a tempo factor of -1.

training, but is important during synthesis to produce smoother transitions between musical contexts (see Section 6 for details). Therefore, we have the following two performance features for each note in the melody:

1. *TempoFactor*: The tempo relative to the performance's average at time, t, and transformed into $\log_2$ space: $\log_2(\mathit{Tempo}_t/\mathit{Tempo}_{\text{avg}})$
2. Δ*TempoFactor*: The difference between this note's *TempoFactor* and the preceding note's *TempoFactor*.

From the score we extract information on both the note and phrase levels. We currently use four note-level features and two phrase-level features. All of the phrase data was obtained from score analyses by two professional musicians. Although we experimented with models representing multiple levels of phrasing, we achieved the best results using only low-level phrase information. More elaborate hierarchical models that can take advantage of additional phrasing levels may be preferable when more data is available. For each note in the score, we include the following features:

1. *RelativeDuration*: The logarithm of the ratio of this note's duration to that of its successor.
2. Δ*Pitch*: The difference in semitones between this note and its successor.
3. *MetricStrength*: An integer describing the inherent metric strength of a note based on its measure position and the current time signature. Notes starting a measure have a metric strength of 1 (strongest) and other notes in the measure have larger values based on a metric hierarchy (Lerdahl & Jackendoff, 1983). In practice we capped the metric strength value so that the few notes with a metric strength weaker than 4 were treated as if they had metric strength 4.
4. *LastNote*: A binary feature that is true only for the last note in the piece. In our earlier work we found that the final notes of performances were often sustained far longer than their scored duration, and the *LastNote* feature helped the model learn this pattern.
5. *PhraseOffset*: A real-valued number between 0 and 1 which describes the relative position of this note with respect to the current phrase (thus the first note in the phrase has a *PhraseOffset* of 0 and the last note's *PhraseOffset* is 1).
6. *PhraseBoundary*: A discrete feature indicating if the note is: the first note of, last note of, or internal to its phrase.

After distilling these features, each note in a score-performance pair is represented by an 8-tuple having five real valued components and 3 nominal ones.

Figure 1 contains the first two bars of the melody line from Chopin's Etude Op. 10 No. 3. Table 2 shows the score feature values assigned to the second note (recall that the first two features are relative to the following note).

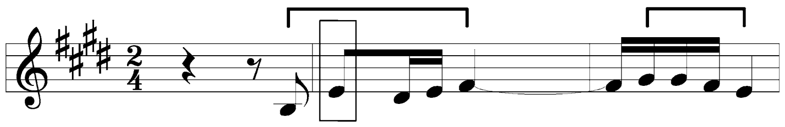

Fig. 1 The first two bars of the melody line from Chopin's Etude Op. 10 No. 3. The brackets above the score give the phrase boundaries used in our experiments. In Table 2 we give the score features for the second note of the melody (contained in the box)

Table 2 Score features computed for the second note of Etude Op. 10 No. 3

RelativeDuration	Δ*Pitch*	*MetricStrength*	*LastNote*	*PhraseOffset*	*PhraseBoundary*
$\log_2(\frac{.5}{.25}) = 1$	-1	1	0	.25	0

4. Hierarchical HMMs for expressive tempo

In this section we briefly review the standard hidden Markov model and then describe the hierarchical Hidden Markov Model learned by the ESP system. A (non-hierarchical) hidden Markov model (HMM) consists of a set of states $Q = \{s_1, \ldots, s_n\}$ that includes a special end state s_e, a *start distribution* over Q, and an output space $\mathcal{O}$. Each non-end state in $Q - \{s_e\}$ contains an output distribution over the space of outputs $\mathcal{O}$ and a transition distribution over Q. The HMM probabilistically generates sequences of outputs as follows. The start distribution is sampled to pick the state for time-step one, call it q_1. State q_1's output distribution is sampled to select the first output, o_1. We say that q_1 *emits* output o_1. The state for time-step two, q_2, is picked by sampling q_1's transition distribution. State q_2 then emits the second output o_2 and picks the state for the third time-step by sampling its transition distribution. This process continues until some q_t makes a transition to the special end state, s_e. At that point the process terminates. HMM's are Markovian because q_t, the state the model is in at time t, depends on the past only through the immediately previous state q_{t-1}. HMM's are "hidden" because all that is observed is the sequence of outputs emitted, and not which state emitted each output.

4.1. Hierarchical hidden Markov models

The hierarchical hidden Markov model (HHMM) was first introduced in 1998 (Fine, Singer & Tishby, 1998). This generalization of the hidden Markov model provides a means of explicitly modeling multi-scale structure in a data set.

Like HMMs, hierarchical hidden Markov models contain a set of states Q and an output space $\mathcal{O}$. Our HHMMs have three kinds of states: *production states*, *abstract states*, and *end states*. These states are organized in a tree structure with the production states at the leaves and the abstract states at the internal nodes. Each group of siblings also contains one special *end state* and the non-end states have a transition distribution over their siblings. Restricting transitions to siblings preserves the hierarchical structure of the model. Thus each individual group of siblings works somewhat like a normal HMM. However, rather than terminating the process, the special *end state* in each group of siblings is used to pass control up the tree.

Furthermore, in an HHMM only the production states contain a distribution over the output space $\mathcal{O}$ and can directly emit symbols. Instead of a distribution over the output space, abstract states contain a start distribution over their children. Each time an abstract state is entered, the HHMM samples the abstract state's start distribution and passes control immediately (in the same time-step) down the tree to the selected child. Control is passed back to the parent after its child end state is entered, this allows the parent to make a transition to one of the parent's siblings. Each entry into an abstract state is considered to generate the entire sequence of outputs emitted by the sub-HHMM rooted at the abstract state. Figure 2 contains a diagram representing an HHMM. Since the root is only a start distribution over its children it is common to omit it from diagrams.

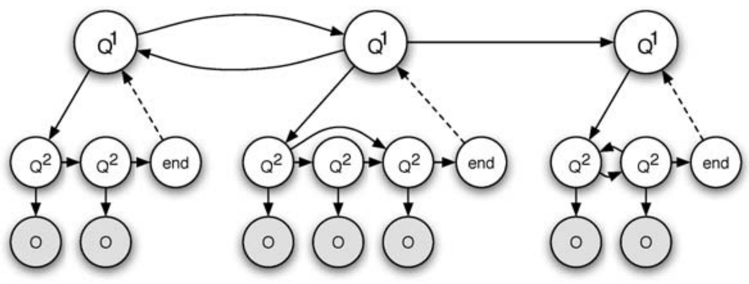

Fig. 2 An example of a depth-2 HHMM (the root is omitted). White nodes represent states in the HHMM while grey nodes represent the outputs emitted by the model

Like an HMM, the HHMM probabilistically generates a sequence of outputs by emitting one output at each time step. At each time step the HHMM will be "in" one state at each level and we use superscripts to denote depths in the tree (with the root at depth 0).

HHMM's can have states organized in a directed acyclic graph rather than a tree, with abstract states sharing sub-models. This sharing reduces the overall model size and can make training easier since the shared structure only has to be learned once. We have not yet been able to exploit this property and use a simple two-level tree structure in our experiments. More elaborate hierarchies may be preferable when more training data is available.

It is important to note that for every HHMM, an equivalent "flat" HMM can be constructed. This is done by creating a separate hidden state for each complete vertical path through the HHMM, so that the one state q_t in the flattened HMM encodes the complete tuple of states $(q_t^1, q_t^2, \ldots, q_t^D)$. Note that the flattened HMM has a special structure as many transitions are impossible and some of the other transitions have constraints on their probabilities. Although an HHMM and its flat HMM counterpart represent equivalent probability distributions over emitted sequences, the loss of hierarchical model structure obscures any multi-scale interpretation of the data. Additionally, any shared model structure in a DAG HHMM will have to be duplicated, increasing the overall model size.

4.2. The ESP model

The ESP system uses a 2-level HHMM for modeling the data. This hierarchy is used to simultaneously learn how score structure and performance variation correlate at two levels of abstraction. Our current architecture has a number of top-level abstract states corresponding to abstract musical contexts at the level of musical phrases. Each of these has a number of production state children representing note level musical contexts. Therefore, the model captures note-level performance strategies conditioned on phrase context. Furthermore, having a separate model for each phrase-level context makes it easier to gain insight from the learned model.

For the experiments reported in Section 7 we use five top-level states each with eight production state children as shown in Fig. 3. We used eight production-level states because eight is the approximate average number of notes per phrase in our data set. The number of top-level states was chosen in an ad-hoc attempt to balance computational considerations and the perceived quality of the synthesized performances. Given the minimal amount of prior information used in determining the model structure, it is likely that it could be further optimized.

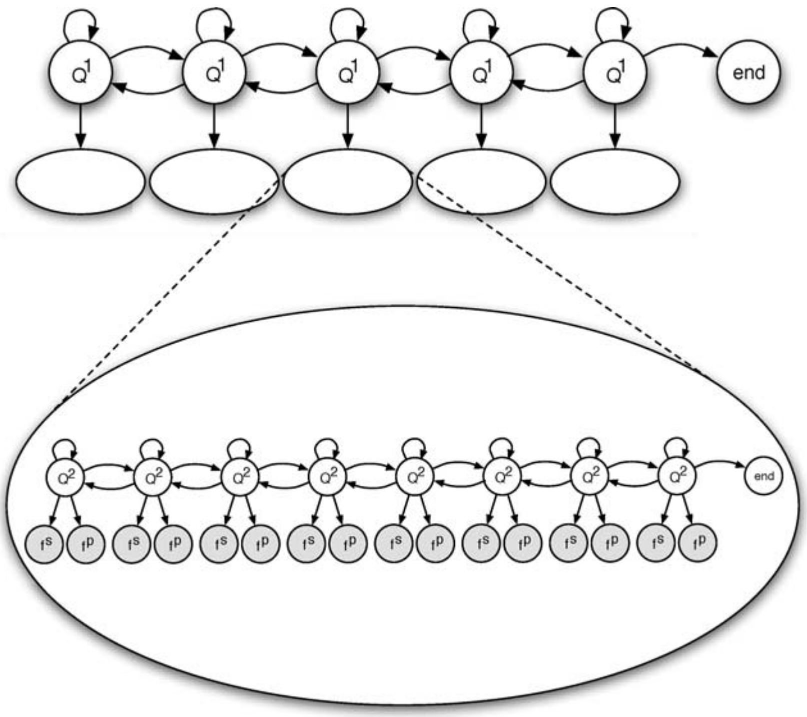

Fig. 3 The basic HHMM used by ESP. There are five phrase-level states, each with eight note-level children. Each group of siblings is ordered left-to-right and is actually fully connected (arcs have been removed to avoid clutter). However, we initialize each state's transition probabilities to decay exponentially with distance.

Our model captures the relationship between score and performance by using a joint distribution over score and performance features in each production state. As discussed in Section 3, tempo change is currently the aspect of expressive performance emphasized by the hierarchical ESP models. (However, the ESP1 system described in Section 8 also represents velocity).

Each production state contains a joint distribution over the eight note-level performance and score features described in Section 3. These joint distributions are encoded as the product of the following five distributions.

1. Two-dimensional Gaussian distribution (with general covariance matrix) over *TempoFactor* and Δ*TempoFactor*.
2. Three-dimensional Gaussian distribution (with general covariance matrix) over *RelativeDuration*, Δ*Pitch*, and *PhraseOffset* features.
3. Multinomial distribution over *MetricStrength* values.
4. Multinomial distribution over *PhraseBoundary* values.
5. Bernoulli distribution over *LastNote* values.

The distributions over score features reflect how applicable the state is to the various notes in the score, and the distribution over performance features guide the tempo of the synthesized notes.

We use a natural initialization of transition probabilities to avoid symmetries within groups of siblings and enforce a correspondence between the hierarchical model and the phrase structure of the score. Each group of production state siblings is linearly ordered. The first state emits the "first" value, the last state emits the "last" value, and the other states emit the "internal" value for the *PhraseBoundary* feature. The transition probabilities are initialized to prefer nearby states in the linear order, falling off exponentially as the distance increases. Transitions between the phrase-level nodes are initialized similarly. Finally, the last production state child of the last phrase-level state is initialized to emit the *LastNote* value true while all other production states emit false. Thus the *PhraseBoundary* and *LastNote* features are deterministically emitted based on the initialization and are not learned from the data.

5. Model training

The goal of model training is to find model parameters $\boldsymbol{\theta}$ that maximize the likelihood that the model generates the observed data. The first step in training is to flatten the HHMM into a (non-hierarchical) HMM. We can then use the Expectation-Maximization (EM) algorithm (Dempster, Laird & Rubin, 1977) to train the flattened HHMM from a set of performances. In the Expectation step the EM algorithm estimates, for each note and production state pair, the posterior probability (given the current model parameters) that the HHMM used that production state to emit the note's features. The maximization step uses these estimates to revise the model parameters in order to increase the likelihood of generating the data sequence. There are several good surveys describing the EM algorithm and how it can be applied to learning HMM models (Bengio, 1999; Bilmes, 1997). We follow Brand in using an entropic estimator (Brand, 1999b) that encourages sparseness when revising the model parameters.

5.1. Computing the expected sufficient statistics

The Expectation step of EM requires us to compute the *occupancy distributions*, the probabilities of being in each state of the HMM at each time-step of each performance in the training data. This process is referred to as *inference*. Murphy and Paskin propose a fast method for exact inference in HHMMs based on treating the HHMM as a dynamic Bayesian network (Murphy & Paskin, 2002). As more data becomes available and the HHMM models become more elaborate, the more powerful dynamic Bayesian network techniques are likely to become increasingly important. However, we found that flattening the HHMM and using the standard forward-backward algorithm (also known as the Baum-Welch algorithm) on the flattened HMM was sufficient for our (2-level) models.

Recall from Section 4.1 that the flat HMM corresponding to an HHMM contains one HMM state for each tuple of states in the HHMM that can be active at the same time (each path from the root to a production state). In our HHMM model we have $n_p = 5$ phrase level parent states each with $n_c = 8$ note level children states, so the corresponding flat HMM has $n_p n_c$ states. This means that the standard forward-backward algorithm can compute the needed probabilities in time $O(n_p^2 n_c^2 T)$ where T is the total number of notes in the training data.

Once it has computed the occupancy distributions, the standard EM algorithm adjusts the transition probabilities and output distributions at each node to maximize the likelihood with respect to the previously computed state occupancy probabilities (Bengio, 1999; Bilmes, 1997). Instead of performing this straightforward optimization we incorporate an entropic prior favoring sparseness in the transitions.

5.2. The entropic MAP estimator

We use a modified version of the Expectation-Maximization algorithm to train the multinomial state transition parameters in our model. This replacement for the maximization step of EM is called the *entropic estimator* (Brand, 1999b). The entropic estimator incorporates the belief that sparse models (those whose parameter distributions have low entropy) are generally preferable to complex models. This bias is motivated by the fact that sparse models tend to be more resistant to the effects of noise and over-fitting, making them better representations of the underlying structure present in the data. We can incorporate this belief into a prior distribution on each set of multinomial parameters, θ using the *entropic prior*.

$$P_e(\theta) \propto e^{-H(\theta)} = e^{\sum_i \theta_i \log \theta_i} = \prod_i \theta_i^{\theta_i} = \theta^\theta \qquad (1)$$

where $H(\theta) = -\sum_i \theta_i \log \theta_i$ is the *Shannon entropy* of the multinomial distribution, θ. While classic EM tries to maximize the likelihood of the model parameters θ given the data $\mathcal{X}$, $\mathcal{L}(\theta \mid \mathcal{X})$, the *entropic estimator* tries to find the maximum *a posteriori* distribution formed by combining the model likelihood with the entropic prior. We can formulate this posterior using Bayes' rule as follows:

$$\begin{aligned} P(\theta \mid \mathcal{X}) &= \frac{P_e(\theta)\mathcal{L}(\theta \mid \mathcal{X})}{P(\mathcal{X})} && (2) \\ &= \frac{e^{-H(\theta)}\mathcal{L}(\theta \mid \mathcal{X})}{P(\mathcal{X})} && (3) \\ &\propto e^{-H(\theta)}\mathcal{L}(\theta \mid \mathcal{X}) && (4) \end{aligned}$$

The proportionality in Eq. (4) is justified because $P(\mathcal{X})$ does not depend on θ and is therefore a constant. It is often more convenient to work with the log-posterior.

$$\begin{aligned} \log P(\theta \mid \mathcal{X}) &= \log P_e(\theta) + \log \mathcal{L}(\theta \mid \mathcal{X}) - \log P(\mathcal{X}) && (5) \\ &= -H(\theta) + \log \mathcal{L}(\theta \mid \mathcal{X}) - \log P(\mathcal{X}) && (6) \end{aligned}$$

The relative importance of the prior and the data likelihood can be adjusted with a trade-off parameter η. Furthermore, since $P(\mathcal{X})$ is independent of θ, maximizing the log-posterior (with the trade-off factor) is equivalent to maximizing the following entropic estimator:

$$-\eta H(\theta) + \log \mathcal{L}(\theta \mid \mathcal{X}). \qquad (7)$$

When $\eta = 0$, the entropic estimator becomes equivalent to EM. An $\eta < 0$ encourages maximum structure by rewarding entropy in an effort to model the data as fully as possible. This setting is useful in the early stages of an annealing schedule (discussed below in Section 5.4). When $\eta > 0$ (specifically when $\eta = 1$, see (Brand, 1999b) for details) minimum entropy solutions are encouraged and model complexity is punished. Varying η during training, typically from $\eta < 0$ up to $\eta = 1$, can help to keep the model parameters from getting stuck in local minima (again, see Section 5.4 for details).

Deriving the entropic estimator is somewhat involved, so we refer the reader to Brand (1999b) for details. Given the expected sufficient statistics (empirical probabilities of the multinomial outcomes) ω, the entropic estimator $\hat{\theta}$ for a multinomial distribution θ is:

$$\hat{\theta}_i = -\frac{\omega_i}{\eta W\left(-\frac{\omega_i}{\eta} e^{1+\lambda}\right)} \qquad (8)$$

where W is the *Lambert W* function (Corless et al., 1996) and λ is a Lagrange multiplier ensuring that the $\hat{\theta}_i$ sum to one. There is a fast iterative method for simultaneously solving for λ and the $\hat{\theta}_i$ (Brand, 1999b).

5.3. Parameter trimming

As with most learning algorithms, graphical models can benefit tremendously from prior knowledge. Significant increases in model quality can often be achieved by encoding information about the problem domain into the model structure.[5] In our case, we can impose some structural constraints (like forcing the start and end of phrases in certain states), but otherwise we are ignorant about how the model should be structured. This leaves us with little choice but to fully connect both the production states under each phrase-level abstract state as well as fully connect the phrase-level abstract states themselves. This is an undesirable situation when training with EM. Because EM only tries to maximize data likelihood, it will make use of the full dynamic range of the state space to model noise as well as other aspects of the data that are not statistically well supported. Thus, fully connected models (with an adequate number of states) that have been trained using EM are likely to overfit, be overly complicated, and generalize poorly.

The entropic estimator addresses these issues with its bias towards "informative" parameter settings. However it also provides a way to roll structure learning into the training process. By introducing trimming criteria into the training cycle, we can begin with an overly-general (i.e. fully connected) model and let the data determine an appropriate structure.

Trimming works by exploiting the fact that we are working to increase the log-posterior of the model (Eq. (7)) rather than just the log-likelihood of the training data alone. Therefore, we can afford to remove a parameter (possible transition) if the reduction of entropy outweighs the decrease in log-likelihood. Using the notation $A_{i\to k}$ for the probability that state i transitions to state k, $A_{i\to *}$ for the vector of transition probabilities out of state i, and $A_{i\to \not{k}}$ for the vector of transition probabilities out of state i to any state other than k, we now present the trimming criteria used in ESP for transition parameters.

$$\eta(H(A_{i\to *}) - H(A_{i\to \not{k}})) > \log \mathcal{L}(A_{i\to *} \mid \mathcal{X}) - \log \mathcal{L}(A_{i\to \not{k}} \mid \mathcal{X}) \qquad (9)$$

Inequality 9 compares the reduction in entropy to the reduction in log-likelihood (with trade-off parameter η) when the transition from state i to state k is removed.

[5] Because EM can never revive a parameter once it has zeroed out, setting initial transition or observation probabilities to 0 provides model structure.

Following (Brand, 1999b), we then approximate the right-hand side of Eq. (9) as follows.

$$\log \mathcal{L}(A_{i\to *} \mid \mathcal{X}) - \log \mathcal{L}(A_{i\to \not k} \mid \mathcal{X}) \approx A_{i\to k} \frac{\partial \log \mathcal{L}(A_{i\to *} \mid \mathcal{X})}{\partial A_{i\to k}} \tag{10}$$

Simplifying the left side of Eq. 9 and using Approximation 10 on the right gives the following trim test:

$$\eta A_{i\to k} \log(A_{i\to k}) > A_{i\to k} \frac{\partial \log \mathcal{L}(A_{i\to *} \mid \mathcal{X})}{\partial A_{i\to k}} \tag{11}$$

$$\log A_{i\to k} < -\frac{1}{\eta} \frac{\partial \log \mathcal{L}(A_{i\to *} \mid \mathcal{X})}{\partial A_{i\to k}} \tag{12}$$

$$A_{i\to k} < \exp\left(-\frac{1}{\eta} \frac{\partial \log \mathcal{L}(A_{i\to *} \mid \mathcal{X})}{\partial A_{i\to k}} \right) \tag{13}$$

The inequality in Eq. (13) provides a cheap test for trimming because the gradient is easily calculated from the state occupancy probabilities computed in the E-step. If the inequality in Eq. (13) holds then removing the transition from state i to state k (setting the transition probability to 0) will increase the optimization criteria in 7 (modulo the approximation).

5.4. Annealing the entropic prior

As mentioned above, traditional EM suffers from the tendency to get stuck in local maxima of the probability space defined by the model parameters.[6] The degree to which this happens is largely a function of the initial parameter settings we assign to the model.

The intuition behind annealing, is that at high "temperatures" (negative values of η in the case of the entropic MAP estimator), we are smoothing the energy surface of the function that we are trying to maximize. This gives the algorithm a better chance of getting into the right "neighborhood" of the function's global maximum.

We can anneal the posterior distribution (Eq. (7)), which the entropic MAP estimator tries to maximize, by changing the weight of the entropic prior. To do this we vary η continuously over the course of all training iterations, typically starting at around -9 and ending at 1. There is little formal theory to guide our choice of "cooling" schedule for η. However, as mentioned previously, $\eta = 0$ and $\eta = 1$, are special cases which correspond to EM and the minimum entropy estimator, respectively. Generally, we begin training with $\eta < 0$ so that the training process begins by exploring many different parameter settings. Depending on our goals for training, we might choose to stop at $\eta = 0$ (if we were more concerned with model fit than generalization) instead of allowing the algorithm to seek the minimum entropy solution.

During the first few iterations of training, we want to give the algorithm a chance to explore the target function's energy surface at a larger scale. Then, as we reduce the temperature, more detail comes into view, allowing us to settle into a (hopefully) global maximum.

[6] This space is also referred to as the *energy surface*, a term born out of simulated annealing's origins in the physics world. We refer to both in our discussions below.

Table 3 Comparison of models produced by training with EM versus those produced by training with the entropic prior. Here ϵ is the smallest positive value represented by the system (MATLAB). Eight models were trained using each training algorithm (with identical starting conditions) and the results were averaged

	Log-likelihood	Perplexity (thresh = .001)	Perplexity (thresh = 2ϵ)
EM	−7336	7.4965	11.3073
Entropic	−7174	6.9427	7.7587

5.5. The benefits of the entropic prior

Brand has conducted several experiments comparing the entropic estimator to EM, with a focus largely on the former's advantages in terms of model complexity (Brand, 1999; Brand, 1999). We performed a quick test to validate the use of entropic prior with trimming and annealing for our system. Empirically, we find that entropic estimation produces better models than EM, both in terms of log-likelihood and model complexity. We trained using both EM and the annealed entropic estimator with parameter trimming on the entire student data set and measured the perplexity (average number of transitions out of each state) and data fit of the resulting models. Using the same random initializations, we trained 8 times with each technique and averaged the results. Because some state transitions may have very small probabilities, we report perplexity results using two different threshold values (the threshold is defined as the minimum probability that a parameter must have in order to be taken into account when computing perplexity). The results given in Table 3 show that entropic estimation not only produces sparser models, but also ones tending to fit the data better.

Surprising as it may seem, the entropic estimator tends to produce superior models. EM techniques are subject to local minima and can set parameters to zero too early in the optimization process. The annealing schedule helps preserve parameters and smooth the likelihood surface early in the optimization process. The value of η becomes positive after a good region is found (hopefully). This has the effect of encouraging model sparsity and reducing over-fitting.

6. Synthesis

After training, the ESP system uses the HHMM model to synthesize renditions of new scores. More precisely, the system creates a sequence of performance features corresponding to the melody notes in the score. We first find a candidate sequence of performance features $\boldsymbol{f^p}$, and then iteratively update $\boldsymbol{f^p}$ using the Expectation-Maximization (EM) algorithm.

6.1. Generating performance features

Let $\mathcal{M}$ be the learned model, T be the number of melody notes in the new score, and $\boldsymbol{f^s}$ the sequence of score features corresponding to the new score (created in the same way as the score features were created from the training score). It is easiest to describe performance synthesis in terms of the flattened HHMM, so the following treats the model $\mathcal{M}$ as a (non-hierarchical) hidden Markov model.

To create the initial sequence of performance features the system finds the most likely state sequence $\boldsymbol{q} = \langle q_1, \ldots, q_T \rangle$ for the score sequence $\boldsymbol{f^s}$ in the trained model $\mathcal{M}$. Thus state sequence $\boldsymbol{q}$ maximizes the probability $P(\boldsymbol{q}, \boldsymbol{f^s} \mid \mathcal{M})$ ignoring the emitted performance features, and is a kind of Viterbi path through $\mathcal{M}$. We then take each state q_t in $\boldsymbol{q}$ and sample from its performance feature output distribution $P(f^p \mid q_t)$ to get the initial performance features for time t. In principle, we can initialize $\boldsymbol{f^p}$ arbitrarily, but this process gives the EM algorithm a good starting point, speeding convergence and reducing problems with local minima. Note that path $\boldsymbol{q}$ need not be the path through $\mathcal{M}$ that maximizes $\boldsymbol{P}(\boldsymbol{f^p}, \boldsymbol{f^s} \mid \boldsymbol{q}, \mathcal{M})$, so further improvement to the synthesized performance is possible using the EM algorithm.

The Expectation-step computes the occupancy probabilities $\lambda_t(i)$ for every state s_i in $\mathcal{M}$ and time $1 \leq t \leq T$. Each $\lambda_t(i)$ is the probability that the model is in state i at time t conditioned on the sequence of outputs, and is easily computed with the forward-backward algorithm.

From the derivation in Wang et al. we get the following Maximization step formula for the f_t^P feature values (Wang et al., 2003).

$$f_t^P = \sum_i \lambda_t(i)\, \mu_i^P\, \Sigma_i^P \left[\sum_i \lambda_t(i)\, \Sigma_i^P \right]^{-1} \qquad (14)$$

where μ_i^P and Σ_i^P represent the mean and covariance matrix for the performance feature distribution in state i of the model.

6.2. Feature smoothing

Recall from Section 3 that there are two real valued performance features for each note, the *TempoFactor* reflecting the relative tempo at the note and the Δ*TempoFactor* measuring the change in relative tempo at the note. Since the human ear easily picks up changes in tempo (and volume), neither feature alone is sufficient to produce good quality renditions. If only the *TempoFactor* is used then the occupancy probabilities at time t can favor states with slow *TempoFactor*s, while the occupancy probabilities at time $t+1$ can favor states with faster *TempoFactor*s. This causes erratic and uncomfortably abrupt tempo changes in the synthesized performance. On the other hand, using only the Δ *TempoFactor* feature results in a different kind of problem. When the occupancy probabilities for several consecutive notes all emphasize the states with high (low) Δ*TempoFactor*s, then the cumulative effect makes the synthesized performance unrealistically fast (slow). Our solution is to combine these two seemingly redundant features in order to produce smooth and reasonable tempo variations.

Assume that we are about to synthesize the tempo for a particular note using the tempo at the previous note and the distribution over *TempoFactor* and Δ*TempoFactor* in a particular production state. The natural approach is to assign the note a tempo such that the joint probability of the resulting *TempoFactor* and Δ*TempoFactor* is maximized. This enforces a compromise between the redundant (and perhaps conflicting) features with the covariance matrix of the distribution influencing the degree to which each feature compromises.

The situation when synthesizing a performance is more complicated since the previous note's tempo can be adjusted if it improves the overall fit for the whole piece. As described by Wang et. al., the second-order features (Δ*TempoFactor*s in our case) can be expanded into differences of first order features just before taking derivatives of the auxiliary Q function in the EM algorithm. This results in a system of linear equations whose solution optimizes the first order features with respect to the second order information.

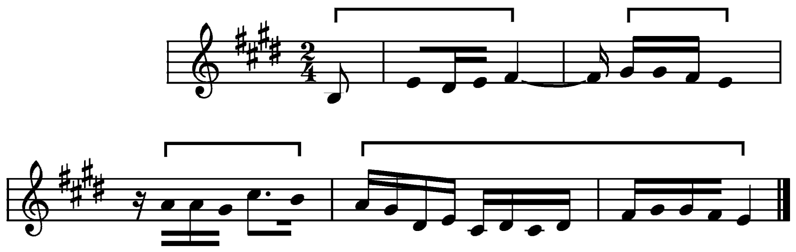

Fig. 4 Score of the melody line adapted from Chopin's Etude Op. 10 No. 3 and used as test data for our synthesis experiment

7. Experiments

This section describes our experiments with the ESP system. All experiments were implemented in MATLAB and used the BayesNet (Murphy, 2004) toolbox for some of the machine learning algorithms. Raw experimental results will be available for download at:
`http://www.media.mit.edu/~grindlay/ESP/ESP.html`.

7.1. Synthesizing performances of novel scores

In our first experiment, we compare the results of the ESP synthesis algorithm to real performances. We trained an HHMM on the students' performances of Träumerei and Prelude No. 15 in D♭, eight separate times. Each model, $\mathcal{M}_i$, had 5 phrase-level states with 8 note-level states in each phrase sub-model, giving a total of 40 production states. The start distributions of both the phrase and note-level models were uniform. Transition probabilities were set to fall exponentially off of the diagonals and emission probabilities were initialized randomly (with the exception of the phrase distributions which use the fixed values described in Section 4). The entropic prior was annealed during the training process according to a cooling schedule that increased from -9 to 1 geometrically. We allowed transition parameter trimming on all levels once the temperature rose above 0.1.

Once trained, we used the score from Etude Op. 10 No. 3.(see Fig. 4) to render a synthetic tempo-factor curve, f_i^{synth}, from each model $\mathcal{M}_i$, according to the algorithms described in Section 6. We report on synthetic renderings of the Etude because its notes' features are well-represented in the other scores. In contrast, both Träumerei and Prelude contain notes whose score features do not appear in the other two pieces. The resulting tempo-factor curves were then averaged[7] yielding, f_{avg}^{synth}. Next, we computed the average of the 7 tempo-factor curves extracted from the student performances of Etude Op. 10 No. 3, yielding f_{avg}^{stud}. We then computed the correlation, mean square error (MSE), and mean absolute error (MAE) of f_{avg}^{synth} relative to f_{avg}^{stud}. We report these three measures since it is unclear how one can reasonably quantify the perceptual similarity of performances. In order to provide a basis for comparison, we also computed the MSE and MAE of a flat tempo-factor curve, f^{flat}, relative to f_{avg}^{stud}. This flat curve represents a performance with no variations in tempo.

[7] The individual tempo-factor curves showed some variation, more than one would expect from the similar likelihoods of the trained models. This leads us to suspect that we did not have enough data to fully overcome the random initialization of emission probabilities, and thus averaging several trained models is a reasonable approach. It is likely that better results could be achieved if either more training data were available or the emission probabilities were initialized to musically sensible values.

Table 4 Comparison of synthesized and flat tempo curves relative to human performance

	MAE	MSE	Correlation
f_{avg}^{synth} vs. f_{avg}^{stud}	.1265	.0200	.5045
f^{flat} vs. f_{avg}^{stud}	.1060	.0169	–

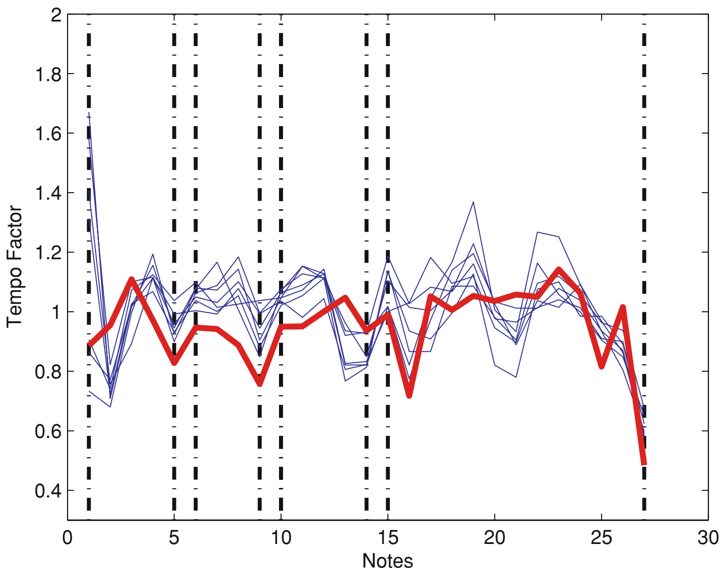

Fig. 5 The average synthetically generated tempo-factor curve and the performances of 7 student performers. The vertical lines denote phrase boundaries

Figure 5 shows the synthetic tempo-factor curve along with the tempo-factor curves of the 7 student performers. As indicated in Table 4, the average synthetic curve was reasonably correlated with the average student performance, but did not perform as well as the flat curve under the average and squared error metrics. This is likely the result of our relatively small training set, although differences in student performance style may have played a part as well. Despite the somewhat disappointing error levels, the synthetic curve does have some interesting properties. Perhaps most importantly, the synthetic curve follows the phrase structure in the score. Other researchers have shown that the tempo curves of individual phrases can generally be well described by second-order polynomials (Todd, 1992; Repp, 1992). From Fig. 5 we can see that the synthetic tempo curve contains (roughly) concave subsections that align nicely with the phrasing of the score. These phrase shapes agree with the general trends of the student performances.

It is worth noting that other researchers using nearest neighbor techniques found expressive piano tempo much harder to synthesize than dynamics (Tobudic & Widmer, 2003). Their methods, experimental setup, and data (performances of Mozart piano sonatas) are very different from ours, making a direct fair comparison difficult. Although they had excellent results with dynamics, many versions of their algorithm often performed worse than a constant tempo synthesis. They report correlation factors ranging from 0.17 to 0.81 (with averages for different algorithms ranging from 0.28 to 0.38). The correlation factor between the tempo synthesized by ESP and the average of the student performances is greater than 0.5.

Table 5 Comparison of HHMM and HMM synthesized tempo curves relative to human performance

	MAE	MSE	Correlation
f^{HHMM} vs. f^{stud}_{avg}	.1265	.0200	.5045
f^{HMM} vs. f^{stud}_{avg}	.2119	.0616	.0739

7.2. Comparison of HHMMs and HHMs

There are several reasons why a hierarchical approach to performance modeling is beneficial. Aside from potential advantages in examining trained models for musicological insight, it is our contention that HHMMs are better suited to the synthesis task than are flat HMMs. We tested this belief by directly comparing the synthesized tempo curves produced by a flat HMM with those produced by the HHMM described above in Section 7.1. We used the same initial starting conditions and the same number of total states in both models. We trained eight HMMs on the same student data used with the HHMM, synthesized eight tempo curves of Etude Op. 10 No. 3, and then averaged them. The resulting tempo curve was then compared with the average student performance in the same manner as described in Section 7.1. A comparison of the HMM and HHMM average results is given in Table 5.

We can see that not only are the MAE and MSE values significantly lower for the HHMM generated curve, but the HHMM generated curve has a much higher correlation coefficient than the curve generated by the HMM. This provides evidence that the hierarchical form allows the model to better capture the multiscale structure present in the tempo and score data.

7.3. Recognizing skill (or Lack Thereof): Professional vs. student pianists

It is generally accepted that expert performers exhibit a greater degree of stylistic individuality than lesser skilled performers do. However, there is also evidence that, despite individual differences, the average performance profile of professional performers is quite similar to that of student performers (Repp, 1997). This suggests that a common performance standard exists for performers of varying skill level.

One of the advantages of probabilistic models, such as ESP, is that they can provide quantitative answers to questions of data and model similarity. In this experiment, we sought to compare student and professional performances both individually and as groups. We trained a separate model, θ_i^{prof}, on each of the 28 professional performances of Träumerei, f_i^{prof}, and a separate model, θ_i^{stud}, on each of the 10 student performances of Träumerei, f_i^{stud}. We then computed separate log-likelihood values for each performance under each of the 38 models. Finally, we computed the means and standard deviations of the average log-likelihoods of performances under each model. This was done for each combination of model and performance class:

$$\mathcal{LL}(\theta_i^{prof} \mid f^{prof}) = \frac{1}{28}\sum_{j=1}^{28} \log P(f_j^{prof} \mid \theta_i^{prof})$$

$$\mathcal{LL}(\theta_i^{prof} \mid f^{stud}) = \frac{1}{10}\sum_{j=1}^{10} \log P(f_j^{stud} \mid \theta_i^{prof})$$

$$\mathcal{LL}\left(\boldsymbol{\theta}_i^{stud} \mid \boldsymbol{f}^{stud}\right) = \frac{1}{10}\sum_{j=1}^{10} \log P\left(\boldsymbol{f}_j^{stud} \mid \boldsymbol{\theta}_i^{stud}\right)$$

$$\mathcal{LL}\left(\boldsymbol{\theta}_i^{stud} \mid \boldsymbol{f}^{prof}\right) = \frac{1}{28}\sum_{j=1}^{28} \log P\left(\boldsymbol{f}_j^{prof} \mid \boldsymbol{\theta}_i^{stud}\right)$$

The summary statistics are given in Table 6. Figure 6 shows the log-likelihoods under the student models while Figure 7 shows log-likelihoods under the professional models.

As can be seen in Table 6 and Fig. 6 and 7, not only are the standard deviations significantly higher under the professional models, but the mean log-likelihood of professional performances under professional models is actually *lower* than it is for professional performances under student models. Additionally, the mean log-likelihood of student performances under student models is significantly higher than the mean of professional performances under professional models. Collectively, these results support the claim that professional performance style varies significantly more than does student performance style.

Table 6 Summary statistics for the log-likelihood values of student and professional performances under student and professional models

	$\mathcal{LL}(\boldsymbol{\theta}_i^{prof} \mid \boldsymbol{f}^{prof})$	$\mathcal{LL}(\boldsymbol{\theta}_i^{prof} \mid \boldsymbol{f}^{stud})$	$\mathcal{LL}(\boldsymbol{\theta}_i^{stud} \mid \boldsymbol{f}^{stud})$	$\mathcal{LL}(\boldsymbol{\theta}_i^{stud} \mid \boldsymbol{f}^{prof})$
Mean	−437.8122	−358.5902	−295.8087	−421.4574
Std. dev.	51.0898	63.0138	35.3700	39.5231

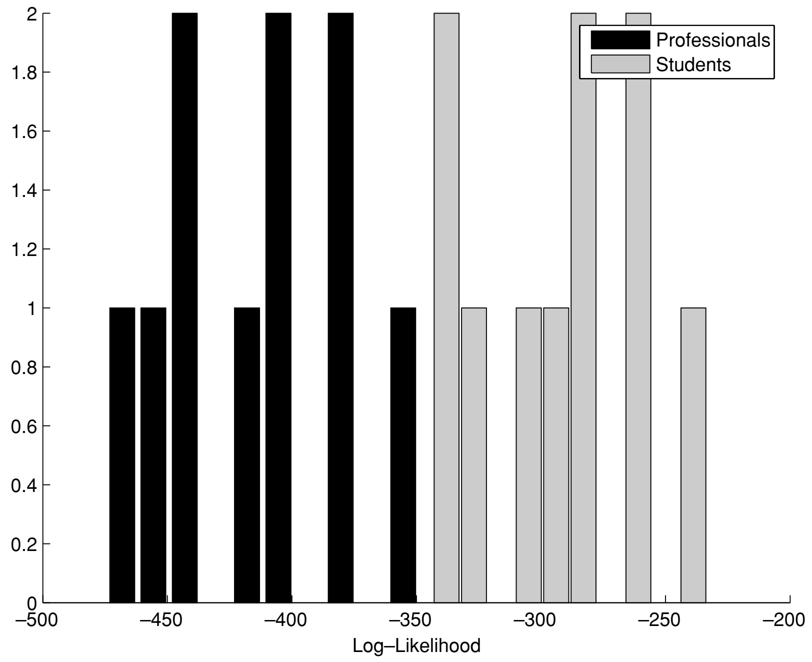

Fig. 6 Histogram of the averaged log-likelihood values for professional and student performances under the 10 student models

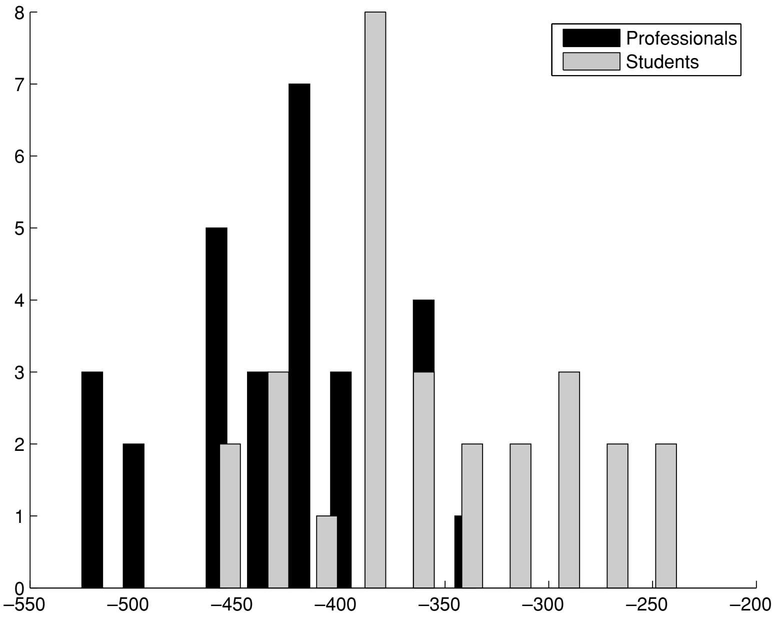

Fig. 7 Histogram of the averaged log-likelihood values for professional and student performances under the 28 professional models

Although these summaries help us to understand the general differences in student and professional model variance, they don't provide information about the global relationships between individual performers. In an effort to gain some insight into these relationships, we formed a matrix of log-likelihood values for each of the 38 performances (columns) under each of the 38 models (rows). We then zeroed the diagonal by subtracting the diagonal elements from their corresponding rows and then added the resulting matrix to its transpose. The resulting matrix M gives a dissimilarity measure on performances. We performed a non-metric multidimensional scaling of M down to two dimensions. The result is shown in Fig. 8.

Figure 8 provides an interesting view of how the performers cluster. Again we observe a significantly larger model variance for the professionals than for the students. We can also see that the student models cluster fairly close to the centroid of the professional models, suggesting that the student performances are very close to the average of the professional performances. This agrees with the findings of Repp (1997).

7.4. Recognizing individual performers

7.4.1. Professionals

A related, but perhaps more challenging problem than performer skill classification, is the recognition of individual performers. Recall that our professional data set contains three performances by both Alfred Cortot and Vladimir Horowitz. To test whether or not ESP was capable of recognizing an unseen performance of Träumerei by a specific pianist, we conducted two separate experiments, one for Cortot and one for Horowitz. In each experiment,

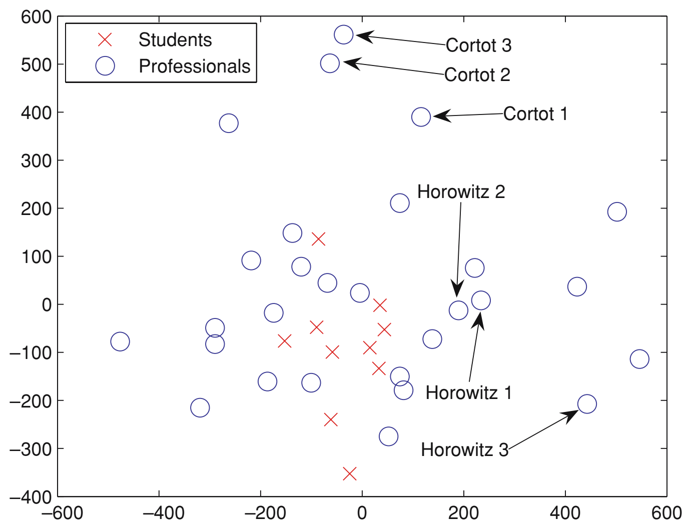

Fig. 8 A 2D multidimensional scaling of the symmetric differences between individual performance models

we trained a model on two of the performer's performances and used the third performance as the test piece. Then we computed the log-likelihood of the test performance as well as the performances of all other professional performers. Each experiment was three-fold cross validated by running the experiment three times, once for each permutation of training and test performances.

For the Cortot recognition experiment, the results were quite good. The model succeeded in recognizing (assigning the highest log-likelihood) to the Cortot test performance in all three cases. This is not surprising given that Cortot's models occupy a fairly isolated area of the dissimilarity space in Fig. 8.

The results of the Horowitz recognition experiment were somewhat mixed. In this case, the model correctly recognized the test performance in one of the three cases, assigned the third highest score in another of the cases, and assigned the fifteenth highest score in the third case. Again, looking at Fig. 8, this is not terribly surprising. Although two of the Horowitz performance models appear to be very similar, they are also similar to several other professional performance models. While Horowitz's third performance model is relatively isolated from the other performer's performance models, it is also fairly dissimilar to his other two performances.

7.4.2. Students

Given that our results up to this point generally seem to indicate that students tend to have more similar performance styles than professionals do, it seems reasonable to expect that student performer recognition would be a more difficult task than professional performer recognition. We tested this idea by running the same set of experiments as described in Section 7.4.1, but on our set of student performances of Prelude No. 15 in D♭. Because two of the seven performers in this set, ALT and PYL, each had recorded three usable takes of Prelude No. 15 in D♭, we used them in the experiment. As before, two separate experiments were run, one for each performer. Each experiment was three-fold cross validated by running it three times, once with each of the performance takes rotated in as the test performance.

The results of the student performer identification task were very good. The model correctly identified ALT's test performance in each of the three experimental runs. Interestingly, in all three runs, ALT's third performance received higher log-likelihood scores than ALT's other two performances, even when it was the test performance.

PYL's identification results were also good. ESP correctly identified the performer in one of the three experimental runs. In one of the other runs, the test performance received the second highest score (and was within 3% of the highest log-likelihood). The third run was particularly interesting because not only did the test run receive the second highest score, but the top-ranked performance actually scored higher than one of the performances on which the model was trained.

There are several possible reasons why our student recognition results were better than one might have expected. First, the set of performances being scored was much smaller than in the professional performer recognition experiment ($16 - 2 = 14$ performances in the student recognition experiment versus $28 + 10 - 2 = 36$ performances in the professional recognition experiment). Second, Prelude No. 15 in D♭ may have more interpretative "room" than Träumerei. Finally, not only were the student performances recorded over a short period of time (unlike the professional performances), but the relative lack of individual style might have the effect of reducing the variation between their performances.

8. Synthesizing tempo and dynamics with the original ESP system

The ESP tempo system described in the previous sections is actually the second incarnation of the ESP system. Before we obtained the professional data set we built an earlier version of ESP specifically for the richer student data set. To avoid confusion we will call the previously described system *ESP* and the version described in this section *ESP1*. Since the student data set contains velocity (loudness) information, the ESP1 system uses (and synthesizes) additional performance features. We conducted a listening test with the ESP1 system to compare its synthesized performances with the melody line produced by student performers. Here we briefly describe how ESP1 differs from ESP and the results of our experiment. See Grindlay's thesis (Grindlay, 2005) for further details on the ESP1 system.

8.1. The ESP1 system

ESP1 uses standard HMMs rather than the hierarchical HMMs. The features used by ESP1 are similar in spirit to those described in Section 3. However, the additional detail in the student data enabled us to use first- and second-order velocity features as well as additional timing features.

We also tried pitch and articulation score features and articulation performance features, but they did not seem to improve the quality of the synthesized performances. With the limited size of the student data set increasing the number features diluted the model's ability to generalize.

8.2. Listening test

To test the ESP1 system we performed the following informal listening test. An HMM was trained on 10 performances in the student data set of Schumann's Träumerei. This HMM

Table 7 Summary results of an informal listening test where 14 undergraduate music students ranked three versions of 10 bars (melody only) from Chopin's Etude Op. 10 No. 3. Each student heard an inexpressive version rendered literally from the score, ESP1's expressive synthesis, and either a performance by an undergraduate or recent graduate in piano performance and ranked them from best to worst

Times ranked	Inexpressive performance	Human Ugrad + Grad = Total	ESP1 synthesis
Best	2	$1 + 2 = 3$	9
Middle	5	$3 + 5 = 8$	1
Worst	7	$3 + 0 = 3$	4

was then used to generate an expressive synthetic performance of a different piece, the first 10 bars of the melody from Chopin's Etude Op. 10 No. 3. Three other renditions of the 10-bar melody line were recorded as well: a literal synthetic rendition of the score (i.e. no expressive variation), a performance by a local music graduate student, and a performance by an advanced undergraduate majoring in piano performance.[8] Each of the human performers was given a warm-up run and then recorded three times with the best sounding recording (to our ears) used in the experiment. The human performers played only the melody line of the 10 bar segment on a Yamaha Clavinova and recorded in MIDI format with all pedalling data removed.

We arbitrarily selected fourteen music students from the halls of the music building to participate in the listening test. Each subject listened to and ranked three performances: the expressive synthetic, the inexpressive synthetic, and one of the two human performances (without knowing which was which). The performances were presented in random order, and the listener could re-hear performances as often as desired. The frequencies of each performance/ranking pair are given in Table 7.

Although far from conclusive, the experiment did show some interesting trends. First, although some listeners (4 of 14) preferred the inexpressive score to the expressive synthesis, most (10 of 14) ranked the expressive synthesis higher. The expressive synthesis also was clearly preferred over the undergraduate's performance (5 of 7 times). The expressive synthesis seemed at least comparable (preferred 4 of 7 times) to that of the graduate student, who presumably has a similar skill level as the students producing the training data. This gives modest evidence that our system is successfully able to synthesize performances of novel scores at a skill level approximating that of the performers generating the training data.

9. Conclusions

The problem of synthesizing expressive performance is both exciting and challenging. Musical performance is one of the many activities that trained people do very well without knowing exactly how they do it. Other examples abound from the physical tasks of catching a ball and driving a car to the more cognitive tasks of reading handwriting, recognizing spoken words, and understanding language and images. Systems that can produce reasonable results for problems such as musical performance are extremely useful not only because of

[8] To be honest, we also recorded a professional pianist playing the Etude, but we judged that her performance was superior to our training data, and so omitted it from the listening tests. One cannot expect an automated system to generate performances better than those it was trained with.

their utility in practical applications, but also because they can help to illuminate the tacit knowledge employed by skilled performers.

The difficulty of producing high-quality data and the scarcity of public data sets makes it extremely difficult to compare and evaluate the different approaches to learning expressive performances. We believe that the hierarchical sequence structure of music is a natural match for hierarchical hidden Markov models and that our results provide good evidence for the effectiveness of this approach. The use of entropic estimation helps produce sparser and better models, and its use in other applications should be considered. We originally intended to use multiple levels of phrasing in our models, but the simple two-level models reported here performed at least as well in our preliminary experiments. We expect that more complex models will be better suited to larger and more diverse data sets.

Our initial approach to performance synthesis addresses only a small part of this difficult problem. Even with an instrument like the piano with relatively simple dynamics there are important aspects that we ignore. Consideration of both melody and harmony presents a number of new challenges because of the dramatic increase in performance variation possibilities. Although we have ideas on how to represent some of the additional dimensions (like the "rolling" of chords), we do not yet have a good grasp of all the important aspects requiring additional features. Even in piano melodies we ignore potentially important performance aspects like staccato and legato, grace notes, and the use of the pedals in order to focus on the more fundamental tempo and dynamics issues. Although we have tackled only a portion of the "real" problem, our techniques are easily generalized and can scale up as more and richer data becomes available.

Given our restricted focus and limited data we have achieved reasonable results when synthesizing performances of new scores. The listening test indicates that people rank ESP1's performance at least comparably to that of pianists having (approximately) the same skill level as the performers producing the data. More quantitative comparisons with human performances show that ESP learns and reproduces much of the tempo variation structure used by trained pianists (see Fig. 5).

The generative models learned by ESP have uses beyond synthesis. For example, our experiments showed that they were often surprisingly good at recognizing individual performers. The various sub-models provide a concise representation of performance strategy that could be interpreted by a human expert to gain new musicological insight. These models provide confirmation of earlier findings (Repp, 1997) regarding the degree of expressive variation employed by professional and trained non-professional performers. In combination with projection techniques such as MDS, the ESP modeling scheme provides an intuitive as well as quantitative similarity measure for both performers and performances. Finally, models like those produced by ESP would be very useful for a number of musical information retrieval tasks such as performance search, classification, and recommendation systems.

Acknowledgments We thank Bruno Repp for generously allowing use to work with his performance data as well as Todd Marston and Tev Stevig for helping with score analysis. We also greatly appreciate the many helpful comments and suggestions made by the special issue editor and the anonymous reviewers.

References

Arcos, J., & de Mántaras, R.L. (2001). An interactive cbr approach for generating expressive music. *Journal of Applied Intelligence*, *27*(1), 115–129.

Bengio, Y. (1999). Markovian models for sequential data. *Neural Computing Surveys*, *2*, 129–162.

Bilmes, J. (1997). A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models. Technical Report ICSI-TR-97-021, University of California at Berkeley.

Brand, M., & Hertzmann, A. (2000). Style machines. In: *Proceedings of ACM SIGGRAPH 2000* (pp. 183–192).

Brand, M. (1999a). An entropic estimator for structure discovery. In M. J. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems 11*. MIT Press: Cambridge, MA.

Brand, M. (1999b). Pattern discovery via entropy minimization. In: D. Heckerman and C. Whittaker (Eds.), *Artificial Intelligence and Statistics*, Morgan Kaufman.

Brand, M. (1999c). Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, *11*(5), 1155–1182.

Brand, M. (1999d). Voice puppetry. In: A. Rockwood (Ed.), *Proceedings of ACM SIGGRAPH 1999* (pp. 21–28), Los Angeles.

Bresin, R., Friberg, A., & Sundberg, J. (2002). Director musices: The KTH performance rules system. In: *Proceedings of SIGMUS-46*, Kyoto.

Casey, M. (2003). Musical structure and content repurposing with bayesian models. In: *Proceedings of the Cambridge Music Processing Colloquium.*

Cemgil, A., Kappen, H., & Barber, D. (2003). Generative model based polyphonic music transcription. In: D., Heckerman and C. Whittaker (Eds.), *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. New Paltz, NY.

Cemgil, A., & Kappen, H. (2003). Monte carlo methods for tempo tracking and rhythm quantization. *Journal of Artifical Intelligence Research*, *18*, 45–81.

Corless, R., Gonnet, G., Hare, D., & Knuth, D. (1996). On the lambert W function. *Advances in Computational Mathematics*, *5*, 329–359.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum-likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistics Society, Series B*, *39*(1), 1–38.

de Mántaras, R.L., & Arcos, J. (2002). AI and music: From composition to expressive performances. *AI Magazine*, *23*(3), 43–57.

Dixon, S. (2001). Automatic extraction of tempo and beat from expressive performances. *Journal New Music Research*, *30*(1), 39–58.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, *32*(1), 41–62.

Grindlay, G. (2005). Modeling expressive musical performance with hidden Markov models. Master's thesis, Dept. of Computer Science, U.C., Santa Cruz.

Lang, D., & de Freitas, N. (2005). Beat tracking the graphical model way. In: L. K. Saul, Y. Weiss, and L. Bottou, (Eds.), *Advances in Neural Information Processing Systems 17* (pp. 745–752). Cambridge, MA: MIT Press.

Lerdahl, F., & Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press.

Murphy, K.P., & Paskin, M.A. (2002). Linear-time inference in hierarchical hmms. In: T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14* (pp. 833–840), Cambridge MIT Press.

Murphy, K. (2004). The BayesNet toolbox. **URL**: `http://bnt.sourceforge.net`.

Raphael, C. (2002a). Automatic transcription of piano music. In D. Heckerman and C. Whittaker, (Eds.), *Proceedings ISMIR*. Paris. France.

Raphael, C. (2002b). A Bayesian Network for real-time musical accompaniment. In T.G. Dietterich, S. Becker, and Z. Ghahramani, (Eds.), *Advances in Neural Information Processing Systems 14*, (pp. 1433–1439). Cambridge, MA: MIT Press.

Repp, B. (1992). Diversity and commonality in music performance: An analysis of timing microstructure in schumann's trumerei. *Journal of the Acoustical Society of America*, *92*, 2546–2568.

Repp, B. (1997). Expressive timing in a debussy prelude: A comparison of student and expert pianists. *Musicae Scientiae*, *1*(2), 257–268.

Saunders, C., Hardoon, D. R., Shawe-Taylor, J., & Widmer, G. (2004). Using string kernels to identify famous performers from their playing style. In: *Proceedings of the 15th European Conference on Machine Learning (ECML'2004)* (pp. 384–395), Springer.

Scheirer, E. (1995). Extracting expressive performance information from recorded music. Master's thesis, Program in Media Arts and Science, Massachusetts Institute of Technology.

Stamatatos, E., & Widmer, G. (2005). Automatic identification of music performers with learning ensembles. *Artificial Intelligence*, *165*(1), 37–56.

Tobudic, A., & Widmer, G. (2003). Learning to play mozart: Recent improvements. In: *Proceedings of the IJCAI'03 Workshop on Methods for Automatic Music Performance and their Applications in a Public Rendering Contest*.

Tobudic, A., & Widmer, G. (2005). Learning to play like the great pianists. In: *Proceedings of the 19th International Joint Conference on Aritificial Intelligence (IJCAI'05).*

Todd, N. (1992). The dynamics of dynamics: A model of musical expression. *Journal of the Acoustical Society of America*, *91*, 3540–3550.

Wang, T., Zheng, N., Li, Y., Xu, Y., & Shum, H. (2003). Learning kernel-based hmms for dynamic sequence synthesis. *Graphical Models*, *65*(4), 206–221.

Widmer, G., & Goebl, W. (2004). Computational models of expressive music performance: The state of the art. *Journal of New Music Research*, *33*(3), 203–216.

Widmer, G., & Tobudic, A. (2003). Playing mozart by analogy: Learning multi-level timing and dynamics strategies. *Journal of New Music Research*, *32*(3), 259–268.

Widmer, G. (2003). Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intellignece*, 146(2), 129–148.