Automatic Transcription of Piano Music

Christopher Raphael
Univ. of Massachusetts, Amherst
Department of Mathematics and Statistic
Amherst, MA 01003-4515
raphael@math.umass.edu

ABSTRACT

A hidden Markov model approach to piano music transcription is presented. The main difficulty in applying traditional HMM techniques is the large number of chord hypotheses that must be considered. We address this problem by using a trained likelihood model to generate reasonable hypotheses for each frame and construct the search graph out of these hypotheses. Results are presented using a recording of a movement from Mozart's Sonata 18, K. 570.

1. INTRODUCTION

We discuss our work in transcribing sampled recordings of piano music into a MIDI- or piano-roll-type representation. The relevance of this endeavor to the music information retrieval (MIR) community is straightforward: MIDI provides a cartoon-like representation of musical data that is extremely compact when compared to the sampled audio data, yet retains the necessary information for many content based analyses and queries. This work has applications outside of MIR such as score-following for computer-human interactive performance, and is part of a broader research agenda of ours — the musical "signal-to-score" problem.

Our approach is based on hidden Markov modeling since we believe that HMMs [1], [2], and, moreover, the ideas of statistical pattern recognition and statistical machine learning have great potential in music recognition problems. First of all, a given "state of nature," such as a particular voicing of a C major triad, can be realized through a wide range of data (say e.g. spectral) configurations, depending on many variables such as the instrument, the room acoustics, the placement of the microphone, as well as many player-specific variables. This variability of presentation argues for a probabilistic data representation in which one models the *distribution* of the data given the true state of nature. Furthermore, automatically *learning* this probabilistic relationship is inherently more robust and flexible than optimizing a particular model by "hand-tweaking" the many parameters. While our preference for learning models from data is shared by the majority of the machine learning community, we have arrived at this preference not through "received wisdom" but rather due to our significant experience hand-tuning various models. Secondly, musical data is highly structured and any recognition approach is well-advised to exploit rather than disregard this this structure. While musical structure can be partly captured in terms of "rules," an examination of data of reveals that the rules can be more accurately represented as tendencies and are easiest to capture in a probabilistic framework. For this reason we propose learning musical structure by training a probabilistic model from musical score data, rather than imposing any specific assumptions *a priori*. Integral to the HMM methodology is a probabilistic representation of data and the desired interpretation, as well as much need computational machinery for training and recognition.

Early work on automatic music transcription did not emphasize any clear methodological framework and addressed significantly scaled-down transcription problems as in [3], [4], [5]. More recently several authors have demonstrated progress in polyphonic music recognition using so-called "blackboard systems" which attempt to fuse bottom-up and top-down recognition schemes, [6], [7], [8]. [9] describes a system in which prior information is incorporated into a graphical model at multiple levels in the graph's hierarchy, but relies on the computationally intensive simulated annealing method. Certainly among the most successful transcription results are those of Klapuri [10], [11]. Hidden Markov models (and approaches with very similar spirit) have found applications in score following in [12], [13], [14] and [15]. Additionally HMMs have been applied to monophonic music recognition as in [16]. The HMM approach to transcription is formally quite similar to these two applications except that the state space is larger by several orders of magnitude.

2. THE MODEL

Our approach begins by segmenting the acoustic signal into a sequence of frames, each corresponding to about a short "snapshot" of sound. From each of these frames we compute a collection of features which reduce the dimension of the data significantly, while still retaining the necessary information for interpreting the data. We denote these feature vectors as $y_1, \ldots, y_N$. The precise features we use will be discussed later.

Our goal is to assign a label to each of these frames describing the frame's content. The most important aspect of the label is the current collection of sounding pitches, however, we will introduce several "flavors" for each collection in what follows. Our approach is model-based. We form a generative probabilistic model, a hidden Markov model, whose output is the observed sequence of features vectors $y_1, \ldots, y_N$ and whose hidden variables, the labels, correspond to the interpretation we seek.

A hidden Markov model is composed of two process we denote by $X = X_1, \ldots, X_N$ and $Y = Y_1, \ldots, Y_N$. The X process is the "hidden" process or "label" process and describes probabilistically the way a sequence of frame labels can evolve and is assumed to be a Markov chain. Of course, we do not observe the X process directly, but rather observe our feature vector data. The HMM model assumes that the likelihood of a given feature vector depends only on the corresponding label. These assumptions will be made more precise in the following.

2.1 The Label Process

Our goal is to assign a label to each frame of data where each label comes from set of possible values $\mathcal{L}$. The most important component in the label is the pitch configuration or "chord." If we denote the possible notes under consideration as $P = \{p_1, \ldots, p_M\}$ then, in principle, any of the 2^M possible subsets of P, $\mathcal{P}(P)$, correspond to possible chords. Clearly the space of possible chords is enormous. In addition the label contains information that distinguishes between the "attack," "sustain," and "rest" portions of a chord. In particular, if we take $R = \{\text{atck-1}, \text{atck-2}, \text{sust-1} \ldots \text{sust-K}, \text{rest-1} \ldots \text{rest-K}\}$. The label set is then

$$\mathcal{L} = \{(s, r) : s \in \mathcal{P}(P), r \in R\}$$

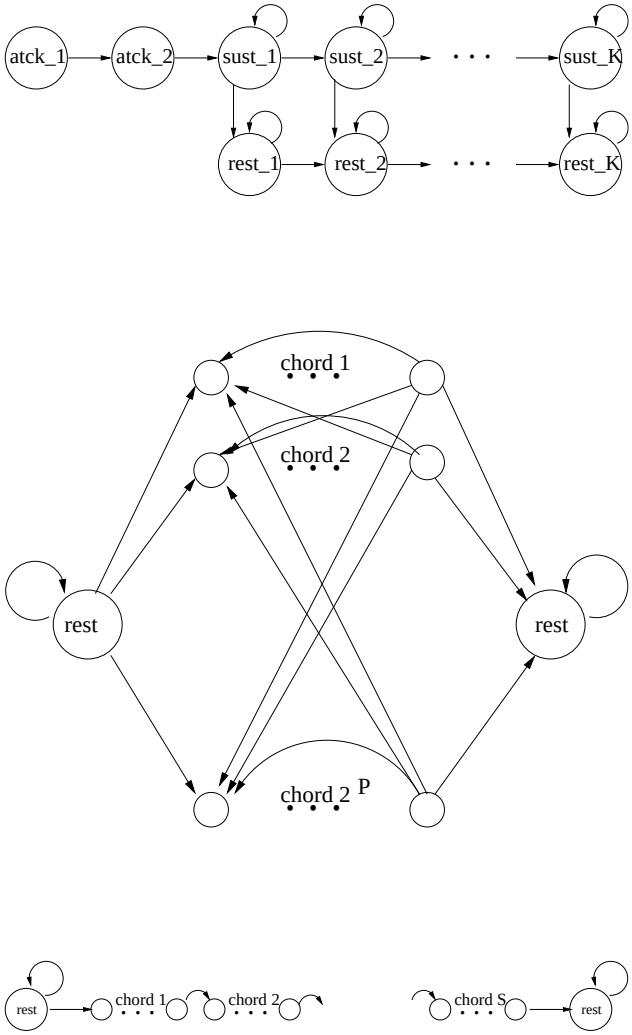

Figure 1: Top: Markov model for a single chord. Middle: Markov model for recognition problem. Bottom: Markov model for training problem.

A more sophisticated model might allow for the association of a flavor (attack, sustain, etc.) for each chord note; in our work here we have opted for the computationally simpler version presented.

We define a random process — a Markov chain — $X_1, \ldots, X_N$ taking values in the label set $\mathcal{L}$. The fundamental assumption of a Markov chain is that the probability of the process occupying a certain state (label) in a give frame depends only on the preceding state (label):

$$\begin{aligned} P(X_{n+1} = x' | X_1^n = x_1^n) &= P(X_{n+1} = x' | X_n = x) \\ &\stackrel{\text{def}}{=} p(x'|x) \end{aligned}$$

where $p(x'|x)$ is the transition probability matrix and X_1^n is defined here by $X_1^n = (X_1, \ldots, X_N)$. We will choose $p(x'|x)$ to incorporate relevant information about the way the label process can evolve. For instance, there is a minimum possible duration for a note. Additionally, certain transitions between chords are more likely than others. The top panel of Figure 1 shows a graph structure depicting the transitions with nonzero probability for a single chord. While the following construction would not be computationally feasible, conceptually the recognition phase of our approach builds such a model for every possible chord and connects the final states of each chord model to the initial state of each chord model. A silence model is constructed to account for the recorded space before and after the performance and are connected as in the middle panel of Figure 1.

Due to the overwhelming size of the state space, an enormous number of possible data interpretations exist. Wherever possible, we have attempted to choose transition probabilities that will guide our recognition toward *a priori* more plausible hypotheses. For instance, we can choose the probability distributions associated with the transitions in the top panel of Figure 1 to reflect a plausible or learned probability distribution on the number of frames spent in a particular note. In addition, we will learn a model for chord transition probabilities from actual score data which we impose on our recognition graph.

2.2 The Observable Process

Rather than observe the label process directly, $x_1, \ldots, x_N$, we observe our feature vector data $y_1, \ldots, y_N$, which are probabilistically related to the labels. The assumption of the hidden Markov model is that each visited state, X_n, produces a feature vector, Y_n, from a distribution that is characteristic of that state. More precisely, we assume

$$\begin{aligned} P(Y_n = y | X_1^N = x_1^N) &= P(Y_n = y | X_n = x) \\ &\stackrel{\text{def}}{=} p(y|x) \end{aligned}$$

thus, given X_n, Y_n is conditionally independent of all other frame labels and all other feature vectors. While the fundamental assumptions of the HMM simplify our situation considerably, the representation and training of the output distributions, $p(y|x)$, is further simplified by two more assumptions.

First of all, recall that we compute a *vector* of features for each frame of data $y = (y^1, \ldots, y^K)$. We assume that the components of this vector are conditionally independent given the state so that $p(y|x) = \prod_{k=1}^K p(y^k|x)$. Furthermore, the states are tied so that many different states share the same feature distributions. That is $p(y^k|x) = p(y^k|T^k(x))$ where the tying function $T^k(x)$ is constructed by hand. Thus we have

$$p(y|x) = \prod_{k=1}^K p(y^k|T^k(x))$$

The nature of the tying function $T^k(x)$ can be clarified by describing more explicitly the features that we compute. First of all, we

need to be able to distinguish between the times when the pianist plays and when there is silence, so we compute a feature y^1 that measures the total energy in the signal. Most the states, $\mathcal{L}$, of our model correspond to various note configurations where we expect the energy content to be relatively high. However both the first and last states of the model, which describe the silence before and after the performance, as well as the "rest" states that represent space that can occur between notes, model situations in which we energy content will tend to be lower. We let $T^1(x) = 0$ for the silence and rest states, and $T^1(x) = 1$ for the remaining states. Thus we must learn two probability distributions: $p(y^k|T^1(x) = 0)$ and $p(y^k|T^1(x) = 1)$ for each of these two cases. The particular values of 0 and 1 are not relevant; the only important issue is the partition of the label set generated by $T^1(x)$: $\{x \in \mathcal{L} : T^1(x) = 0\}$, $\{x \in \mathcal{L} : T^1(x) = 1\}$.

We also need to distinguish between note "attacks" and steady state behavior if we have any hope of detecting rearticulations. For this reason we compute a feature y^2 that measures the local "burstiness" of the signal. Many different measures are possible including differenced energies and differenced spectral energies summed over various regions in frequency space. Our feature y^2 computes several such measures of burstiness (y^2 is a vector). For this feature, states can be partitioned into three groups: the states at the beginning of each note where we expect the signal to have high burstiness, for which we set $T^2(x) = 0$; the states corresponding to steady state behavior where we expect the burstiness to relatively low, for which we set $T^2(x) = 1$; and the silence states where the burstiness measure will have a third type of behavior for which we set $T^2(x) = 2$.

The remainder of our features are concerned with the most fundamental problem of distinguishing between the many possible pitch configurations. Each of the features $y^3, \ldots, y^K$ is computed from a small frequency interval of the Fourier transformed frame data. For each such window we compute the empirical mean and empirical variance when the spectral energy function, restricted to the window, is normalized as a probability distribution. The mean value will estimate the location of the harmonic, (when there is a single harmonic in the window), and the variance can be used to distinguish (probabilistically) between when there is a single harmonic (low variance) and when there is not (high variance). For every configuration of notes we can compute ideally where harmonics will lie in the spectrum. We will tie states that should have similar behavior in a frequency window. For instance, if $x \in \mathcal{L}$ corresponds to a configuration in which none of the notes contain energy in the window indexed by feature k, then $T^k(x) = 0$. Similarly, we have chosen to collect together the states having several harmonics in the window by letting $T^k(x) = 1$ for these states. The remaining states are partitioned by $T^k(x)$ into groups having a single harmonic at approximately the same frequency in the window. That is, the collection of states $\{x \in \mathcal{L} : T^k(x) = t\}$ all have a single harmonic in the kth window at approximately the same location.

Note that we have represented millions of state-conditional distributions in terms of a small number of "basis" distributions.

3. TRAINING THE MODEL

One of the most important virtues of the HMM formulation is that the probability distributions can be trained in an *unsupervised* fashion. This means that we do not need to hand-label frames from all of the different probability distributions described in the previous section. Rather, an iterative procedure, known as the forward-backward or Baum-Welch algorithm, allows us to automatically train from signal-score pairs. Thus, the training assumes that we have a musical score for each sound data file, and therefore we know the approximate sequence of pitch configurations realized in the data, as well as the approximate durations of those configurations. However, we do not have the correspondence between states and frames that would allow a simple-minded training procedure for the output distributions.

When the score is known, we can build a model for the hidden process using the template in Figure 1 (bottom). The algorithm begins from a neutral starting place — we begin with uniformly distributed output distributions — and iterates the process of finding a *probabilistic* correspondence between model states and data frames and then retraining the probability distributions using this correspondence.

More precisely, the forward-backward algorithm allows us to compute the *posterior* distributions $P(X_n = x|Y_1 = y_1 \ldots Y_N = y_N)$. If this distribution concentrated is mass on a single state, x, then it would be reasonable to use the data frame y_n as an example of $p(y|x)$. In the more typical case in which the posterior distribution allow for several possible states, we consider y_n as a fractional example of $p(y|x)$ where the fraction is the posterior probability $P(X_n = x|Y_1 = y_1 \ldots Y_N = y_N)$. The fractional examples are then used to retrain the output distributions by computing empirical probabilities of the various observables.

Our output distributions on feature vectors are represented through decision trees as follows. For each distribution $p(y^k|T^k(x))$ we form a binary tree in which each nonterminal node of the tree corresponds to a question of the form $y^{k,v} \stackrel{?}{<} c$ where $y^{k,v}$ is the vth component of feature k. An observation y^k can be associated with a terminal node by "dropping" the observation down the tree. That is, starting at the root we evaluate the root question and move to the left or right subtree as the answer is yes or no. This process is continued until we arrive at terminal node which we denote by $Q^k(y^k)$. Thus the decision trees quantize the data vectors, and to be precise, the quantized values $\{Q^k(y^k)\}$ are the actual features we use. As the training procedure evolves, the trees are reestimated at each iteration to be progressively produce more informative probability distributions by maximizing the mutual information between the terminal node $Q^k(y^k)$ and the class label $T^k(x)$. Thus we seek to build trees which would function well at predicting the class label $T^k(x)$ of a vector y^k, although we do not use them for this purpose.

4. RECOGNITION

The traditional HMM approach to recognition seeks the most likely labeling of frames, given the data, through dynamic programming:

$$\hat{x}_1^N = \arg\max_{x_1^N} P(X_1^N = x_1^N | Y_1^N = y_1^N)$$

where $\hat{x}_1^N = (\hat{x}_1, \ldots, \hat{x}_N)$. This corresponds to finding the best path through the graph of Figure 1 (middle) where the reward in going from state x_{n-1} to x_n in the nth iteration is given by

$$C_n(x_{n-1}, x_n) = p(y_n|x_n)p(x_n|x_{n-1})$$

A well-known recursion, the Viterbi algorithm, incrementally constructs the optimal paths of length n from the optimal paths of length $n-1$. The computational complexity of full-fledged dynamic programming grows with the square of the state-space which is completely intractable in our case: Under restrictive assumptions on the possible collection of pitches and the number of notes in a chord, the state space is on the order of 10^8. Even with standard pruning techniques, the conventional conception of DP is completely hopeless with a state space so large.

Instead, we use the data model constructed in the training phase to produce a greatly condensed version of the state graph of Figure 1 (middle). For each frame, n, we perform a greedy search that seeks a plausible collection of states $x \in \mathcal{L}$ for that frame. This is accomplished by searching for states x giving large values to $p(y_n|x)$. The search is performed by finding the mostly likely 1-note hypotheses, then considering 2-note hypotheses formed by adding single notes to the best 1-note hypotheses, and so on. Thus

each frame, n, will be associated with a possible collection of states A_n. The states are "blended" by letting $B_n = \cup_{k=n-d}^{n+d} A_k$ represent the collection of states we will entertain at the nth frame. Then the state graph is constructed by restricting the full graph of Figure 1 (middle) to the B_n sets.

The principal disadvantage of such an approach is that if the "true" state at frame n is not captured in B_n, then it cannot be recovered during recognition. However, the approach is computationally feasible. We are currently exploring an extension of the above idea in which, for each frame n, we seek hypotheses $x \in \mathcal{L}$ maximizing $p(x|y_n)$ where

$$p(x|y_n) = \frac{p(y_n|x)p(x)}{p(y_n)}$$

To this end we learn a distribution on states $p(x)$, mostly depending on the chord configuration, from actual score data. Thus the original identification of possible hypotheses is guided toward more plausible configurations. A particularly valuable contribution of this modification is that the "prior" distribution $p(x)$ helps to distinguish between chord configurations we deem "homonyms" — these are chords that are indistinguishable to our data model $p(y|x)$. Examples of chord homonyms are chords differing by the inclusion of an octave or 12th above any of the chord notes.

5. EXPERIMENTS

We present here preliminary experiments on several movements from Mozart piano sonatas. The results presented here are derived from a performance of the 3rd movement of Sonata 18, K. 570. Both the original data and a MIDI file of our recognition results can be heard heard at http://fafner.math.umass.edu/ismir02.

We restricted our attention to notes falling in the true range of the movement, from c two octaves below middle c to f two and a half octaves above middle c. Also consistent with the actual movement, we considered only chords containing four or less notes.

We trained our HMM in the manner described using data also taken from various Mozart piano sonata movements recorded under similar conditions.

While the MIDI file on the web page is likely the easiest description of our level of success to understand, it only allows for subjective comparisons. For this reason we have also performed an objective measure of performance we describe now. Such a measure is essential in developing a recognition system since it allows one to objectively evaluate the consequences of a system modification.

The "Word Error Recognition Rate" is a common measure of system performance used in speech recognition. The essential idea is to compute the minimum "edit distance" — the minimum number of insertions, deletions and substitutions necessary to transform the recognized word sequence into the true word sequence. Recognition error rates are often reported as

$$\text{Error Rate} = 100 \times \frac{\text{Insertions} + \text{Deletions} + \text{Substitutions}}{\text{Total Words in Truth Sentence}}$$

We will use the same convention.

In computing the minimum edit distance we take the sequence of recognized chords ordered by note onset time with each chord spelled in arbitrary order as our recognized data. Thus, for the purposes of computing the error rate, the recognized data is collapsed into a string of notes. Conceptually, we convert the score also into a sequence of pitches, however our "matching" algorithm considers all $n!$ permutations of each n-note chord in the score. Thus, if the score contained a single triad chord, any recognition of two of the triad notes, (as a chord or in any sequence), would produce a single deletion error. This computation can be performed with a small variation on the usual dynamic programming computation of minimum edit distance. The data above yielded a "note error rate" of 39% with 184 substitutions, 241 deletions, and 108 insertions out of 1360 notes.

In computing the recognized sequence of pitches, if two adjacent recognized chords have a pitch in common, it is assumed that the note is not rearticulated. This assumption is correct most of the time and our current model does not have the ability to detect any possible combination of rearticulations. The dominance of deletions in our recognition results is partly due to deletions caused by this simple-minded assumption.

A second significant contributor to both the number of deletions is our inability to distinguish between chord homonyms. In our initial generation of chord hypotheses we consider only the homonym with the least number of notes; thus we do not allow a note to be added to a chord hypothesis if it does not generate any new harmonics. As discussed above, the algorithm cannot recover from errors committed at this stage, due to the need to keep the search space manageable.

6. DISCUSSION

While our recognition results leave room for improvement, it should be noted that our current system was devised as a relatively simple modification of an earlier HMM system used for *monophonic* recognition of a *known* score. Results of this quality might already be useful in a number of Music Information Retrieval applications tolerant of errorful representations.

We expect, however, that some simple additions may yield substantial gains in performance. For instance, the current system works with nearly no knowledge of the plausibility of various sequences of chords. We are currently working on developing a probabilistic model, trained from real music data, that models the likelihood of chord sequences. Such models provide dramatic improvement in other HMM domains. Additionally, our current system makes almost no effort to model the acoustic characteristics of the highly informative note onsets. We hope that a more sophisticated "attack" model would help in recognizing the many repeated notes which our system currently misses.

7. ACKNOWLEDGMENTS

This work is supported by NSF grants IIS-0113496 and IIS-9987898.

8. REFERENCES

[1] Rabiner L. (1989), "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE,* 77, 257–286, 1989.

[2] Bahl L., Jelinek F., Mercer P. (1983) , "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI–(52), 179–90, 1983.

[3] Piszczalski M., Galler B. (1977), "Automatic Music Transcription," *Computer Music Journal*, vol. 1, no. 3, 1977, 24–31.

[4] Moorer, J. (1977), "On the Transcription of Musical Sound by Computer," *Computer Music Journal* vol. 1, no. 4, 1977, 32–38.

[5] Chafe C., Mont-Reynaud B., Rush L., (1982), "Toward an Intelligent Editor of Digital Audio: Recognition of Musical Constructs," *Computer Music Journal* vol. 6, no. 1, 1982, 30–41.

[6] Bello J. P., Monti G., Sandler M. (2000), "Techniques for Automatic Music Transcription," *Proceedings of the International Symposium on Music Information Retrieval*, Plymouth, Massachusetts, 2000.

[7] Martin K. (1996), "Automatic Transcription of Simple Polyphonic Music: Robust Front End Processing," *MIT Media Lab, Technical Report #399*, December, 1996.

[8] Ellis D. (1995), "Mid-Level Representation for Computational Auditory Scene Analysis," *Proceedings of the Computational Auditory Scene Analysis Workshop*, 1995 International Joint Conference on Artificial Intelligence, Montreal Canada, August 1995.

[9] Kashino K., Hagita N., (1996), "A Music Scene Analysis System with the MRF-Based Information Integration Scheme," *IEEE Proceedings of the International Conference on Pattern Recognition*, 725–729.

[10] Klapuri A. (2001), "Multipitch Estimation and Sound Separation by the Spectral Smoothness Principle," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2001.

[11] Klapuri A., Virtanen, Eronen, Seppanen (2001), "Automatic Transcription of Musical Recordings," *Proc. Consistent and Reliable Acoustic Cues Workshop*, Aalborg, Denmark, 2001.

[12] Raphael C. (1999), "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 360–370.

[13] Orio N., Dechelle F., (2001), "Score Following Using Spectral Analysis and Hidden Markov Models," *Proceedings of the ICMC*, 151–154, 2001.

[14] Grubb L., Dannenberg R., (1998), "Enhanced Vocal Performance Tracking Using Multiple Information Sources," *Proceedings of the ICMC*, 37-44, 1998.

[15] Cano P., Loscos A., Bonada J., (1999), "Score-Performance Matching Using HMMs," *Proceedings of the ICMC*, 441-444, 1999.

[16] Durey A., Clements M. (2001), "Melody Spotting Using Hidden Markov Models," *Proceedings of the International Symposium on Music Information Retrieval*, Bloomington, Indiana, 2001.