Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models

Christopher Raphael
Department of Mathematics and Statistics
University of Massachusetts, Amherst
Amherst, MA 01003
raphael@math.umass.edu

Jan 18, 1998

In this paper we address an important step towards our goal of automatic musical accompaniment — the segmentation problem. Given a score to a piece of monophonic music and a sampled recording of a performance of that score, we attempt to segment the data into a sequence of contiguous regions corresponding to the notes and rests in the score. Within the framework of a hidden Markov model, we model our prior knowledge, perform unsupervised learning of the the data model parameters, and compute the segmentation that globally minimizes the posterior expected number of segmentation errors. We also show how to produce "on-line" estimates of score position. We present examples of our experimental results and readers are encouraged to access actual sound data we have made available from these experiments.

Index Terms: automatic musical accompaniment; hidden Markov models, computer music.

1 Introduction

Suppose we are given a musical score consisting of two parts: a solo part and an accompaniment. For example, imagine a sonata for violin and piano. We would like to create a computer program that "listens to" (samples) a live performer playing the solo part and generates the accompaniment in real time. That is, the computer will follow the soloist. The hope is that, through repetition, we can "learn" the soloist's interpretation and that this knowledge can be combined with a real-time score position estimate to produce the accompaniment in a musically satisfying way. Vercoe [1], Vercoe and Puckette [2], Dannenberg [3], Bloch and Dannenberg [4], Dannenberg and Mukaino [5] and Baird et. al. [6] have addressed this automatic accompaniment problem. We also have some preliminary results on this task and the interested reader is encouraged to listen to examples available through a web page given in Section 7.

In this paper we address the more basic *segmentation* problem: Given a musical score for a *monophonic* instrument (one note at a time), and the sampled acoustic signal of a performance of that score, we wish to partition our data into contiguous regions corresponding to the notes and rests (see Fig. 1).

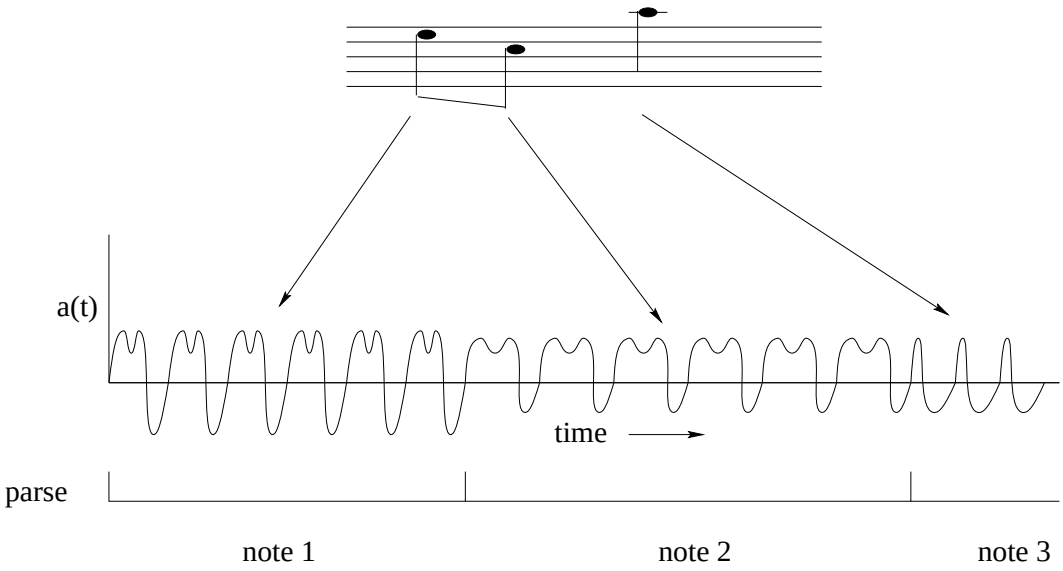

Figure 1: We seek a "segmentation" of the data — a partition of the data into notes and rests.

We call such a partition a segmentation. Our methodology applies, in principle, to any monophonic instrument, and possibly might be extended to small collections of instruments as outlined in Section 8. However, we restrict our attention here to a monophonic source since it is the task we face in automatic accompaniment. We address both "on-line" and "off-line" segmentation here. Off-line segmentation partitions the data using the complete acoustic signal, while on-line segmentation performs this partition as the data is collected. On-line segmentation is the task that relates most directly to automatic accompaniment, however we use similar methodologies in both cases. To focus our presentation, we do not address many of the most important issues in automatic accompaniment here, such as learning the soloist's interpretation and incorporating musical constraints on the performance of the accompaniment.

There are many pitch-detection techniques that classify a short segment of sound as one of the possible musical notes. Generally such techniques rely on analysis of the signal's spectrum or autocorrelation; see, for example, [7]. Such approaches make no use of the score so, although quite general, they are suboptimal for our situation. A possible compromise is to post-processes the output of a "pitch-to-midi" converter, which is a device that transforms an acoustic signal to a stream of pitch labels. For instance, Dannenberg and collaborators [3],[4], [5] analyze the output of the pitch-to-midi converter seeking a best match to the score using dynamic programming techniques. For the most part, pitch-to-midi can provide a workable solution to this problem, however, we believe it is possible to do better. The advantage of dealing with the raw signal is that the recognition scheme can be "tuned" to recognize the particular characteristics of any instrument in any acoustic environment making difficult tasks such as recognizing rearticulations and octave leaps easier. In contrast, pitch-to-midi conversion employs a "one-size-fits-all" approach. In addition, the method we present can be extended to perform simultaneous tracking of multiple instruments from a single audio source. Grubb and Dannenberg [8] present a method for tracking a vocal performer which shares the probabilistic orientation of our approach, although the power of hidden Markov models is not utilized.

In what follows, we present a probabilistic framework — the hidden Markov model — particularly

well-suited to this segmentation problem. Hidden Markov models are most familiar in the literature on speech recognition [9], [10] and our work owes much to that field. We feel that the HMM framework is adaptable to a wide variety of other applications as well such as document decoding [11] and natural language processing [12] to name only two. Within the HMM framework, our *a priori* notion of the plausibility of various segmentations will be described by assigning them probabilities. We then develop a model that describes the likelihood of our acoustic data given a hypothesized segmentation. It is nearly impossible to set the many different data model parameters by hand, so we present a training algorithm that automatically learns these parameters in an unsupervised way. Finally our framework facilitates computationally feasible identification of the *globally* optimal segmentation through dynamic programming algorithms. We define the optimal segmentation to be the one minimizing the expected number of segmentation errors, rather than a segmentation based on the posterior mode and the advantages of this choice are discussed, such as its insensitivity to the choice of discretization. In summary, we demonstrate an approach which integrates flexible modeling, global optimization, and unsupervised learning of unknown model parameters in a natural way.

2 Hidden Markov Models

Our raw data is a sampled version of the acoustic signal generated by the performer which we denote by $a(t)$, $t = 0, 1, \ldots, T-1$. We divide the signal into $K = \lfloor T/J \rfloor$ contiguous blocks of length J called *frames* by

$$y_k(j) = a(kJ + j)$$

for $k = 0, 1, \ldots K-1$ and $j = 0, 1, \ldots, J-1$. We think of the frames, $\{y_k\}$, as "snapshots" of sound, much like the frames of a movie. In our experiments we have used frames corresponding to 32 ms.

An HMM is a stochastic process consisting of two layers — one hidden and one observable. The hidden layer, denoted by $\mathbf{X} = (X_0, X_1, \ldots, X_{K-1})$ is a collection of random variables, (one for each frame), taking values in a state space Σ. Each state is associated with a note in the score, so if we could recover $\mathbf{X}$, then we would know which note was sounding at every frame, thereby segmenting the data. We assume $\mathbf{X}$ is a time-homogeneous Markov chain

$$\begin{aligned} P(\mathbf{X} = \mathbf{x}) &= P(X_0 = x_0) \prod_{k=1}^{K-1} P(X_k = x_k | X_{k-1} = x_{k-1}) \\ &= P(X_0 = x_0) \prod_{k=1}^{K-1} Q(x_k, x_{k-1}) \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_0, x_1, \ldots, x_{K-1})$ with $x_k \in \Sigma$, and where $Q(\sigma_1, \sigma_2)$ is the transition probability matrix which does not depend on the frame k. The transition probabilities and the meanings of the states are described fully in Section 3. We do not observe the $\mathbf{X}$ process, of course, but rather our frame data $\mathbf{y} = (y_0, y_1, \ldots, y_{K-1})$. We assume $\mathbf{y}$ is a realization of a random process $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_{K-1})$ depending only locally on the $\mathbf{X}$ process:

$$\begin{aligned} P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) &= \prod_{k=0}^{K-1} P(Y_k = y_k | X_k = x_k) \\ &= \prod_{k=0}^{K-1} p(y_k | x_k) \end{aligned} \tag{2}$$

where the output distributions, $\{p(y|\sigma)\}$, $\sigma \in \Sigma$, also do not depend on the frame k. We will discuss the output distributions briefly in Section 4 and more fully in Section 6. The assumptions of Eqns. (2) and (3) fully specify the joint probability distribution on our HMM $(\mathbf{X}, \mathbf{Y})$.

Once we set the observable process equal to our frame data, $\mathbf{Y} = \mathbf{y}$, the *posterior* distribution on the $\mathbf{X}$ process is readily computable, so, although we cannot recover the exact hidden state sequence, we can compute $P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y})$. Owing to this fact, a number of good segmentation estimates are possible. The simplest of these is based on the ever-popular posterior mode:

$$\begin{aligned}
\hat{\mathbf{x}} &= \arg\max_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y}) \\
&= \arg\max_{\mathbf{x}} \frac{P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})P(\mathbf{X} = \mathbf{x})}{P(\mathbf{Y} = \mathbf{y})} \\
&= \arg\max_{\mathbf{x}} P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})P(\mathbf{X} = \mathbf{x})
\end{aligned}$$

which is calculated through dynamic programming. Section 5 argues that the segmentation associated with the optimal state sequence might be undesirable. We formulate a more relevant optimality criterion: the (posterior) expected number of segmentation errors. We demonstrate that this notion of optimality leads to readily computable and globally optimal segmentations.

In truth, both of these estimators perform admirably so we choose here partly on philosophical grounds. We have, however, encountered a number of practical situations where our preferred estimate outshines estimates based on the posterior mode. Section 5 presents a case for our preferred estimate and, we hope, calls into question the blatant favoritism enjoyed by the posterior mode in hidden Markov modeling.

3 The Prior Model

Our musical score consists of M notes, each of known length in terms of musical units such as beats or measures. (Grace notes are converted to measured notes). Typically the score also suggests a tempo relating these musical units to a time unit (e.g. 60 beats per minute). The tempo implies an exact or *metronomic* duration for each note, although the difference between theory and practice can be significant.

In this section we describe the Markov chains used to model the individual notes and the manner in which they are connected to form the prior model, $\mathbf{X}$. In addition, each state, $\sigma \in \Sigma$, of the Markov chain $\mathbf{X}$ is "labeled" by a function

$$L : \Sigma \to \Lambda$$

where the label set Λ is

$$\Lambda = \{\text{rest}, \text{pitch}(1), \ldots, \text{pitch}(I), \text{artic}(1), \ldots, \text{artic}(I)\} \tag{3}$$

and the playable pitches are indexed by $1, 2, \ldots, I$. Here $\text{pitch}(i)$ corresponds to the steady-state portion of the ith pitch, while $\text{artic}(i)$ is the attack of the ith pitch. "rest" is, of course, the resting state. The label of a state tells us what we expect to hear when the state is active, although this is made more precise in Section 4.

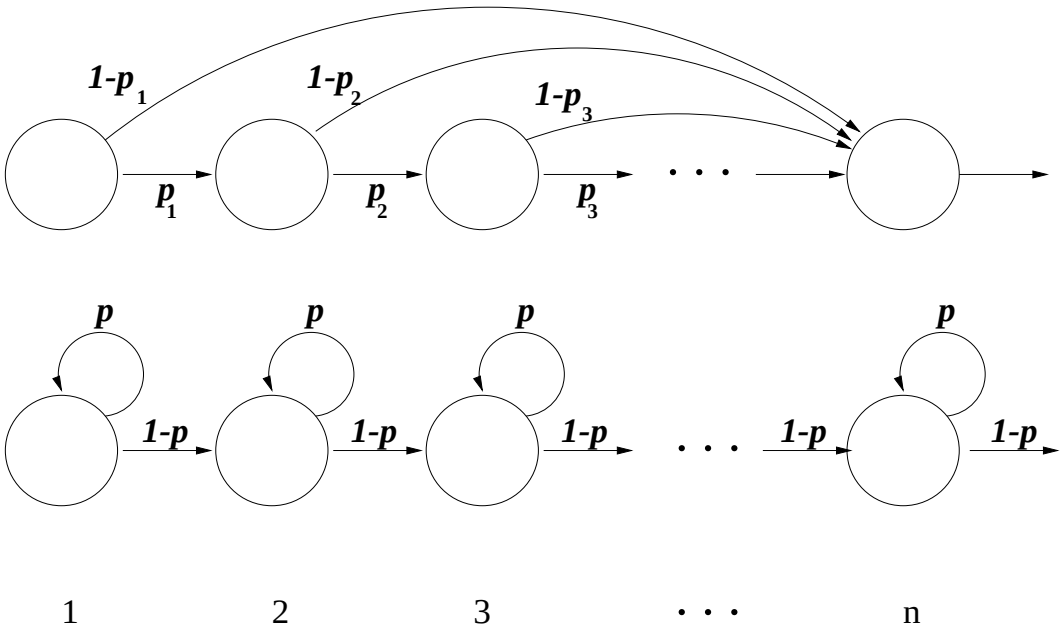

Figure 2: **Top:** A model capable of producing any desired note length distribution. **Bot:** A more parsimonious note length model.

The main issue we address is modeling the lengths of notes. Consider the model in Figure 2 (top). If T is a random variable representing the number of frames spent in a particular note, then any distribution of T can be captured by the model of Figure 2 (top) by setting

$$p_k \quad = \quad P(T > k+1 | T > k)$$

as long as the distribution of T assigns zero probability to T being either less than 2 or greater than the number of states.

Although the model of Figure 2 (top) captures precise information about the length distribution it carries with it a computational burden due to the large number of necessary states. For instance, at typical resolution, it is not at all uncommon to observe notes lasting for more than 50 frames. For such long notes, we generally do not have specific enough information about length distribution to warrant a model using as many parameters as needed in the model of Figure 2 (top). Typically we will not know much more about the length distribution other than its average value and some measure of the variability we observe. The model of Figure 2 (bottom) uses only a few states and leads to a simply parameterized length distribution that can still represent the information about length that we have available. It is readily seen that the distribution on the number of frames spent in such a model is negative binomial. That is

$$P(T = t) = \binom{t-1}{n-1} p^{t-n} (1-p)^n \tag{4}$$

for $t = n, n+1, \ldots$. In practice, we have used the model of Figure 2 (bot) to model longer notes by choosing the parameters n, p to approximate the desired length distribution. Shorter notes are better handled by the model of Figure 2 (top) due to the relatively small number of parameters needed to represent their length distributions.

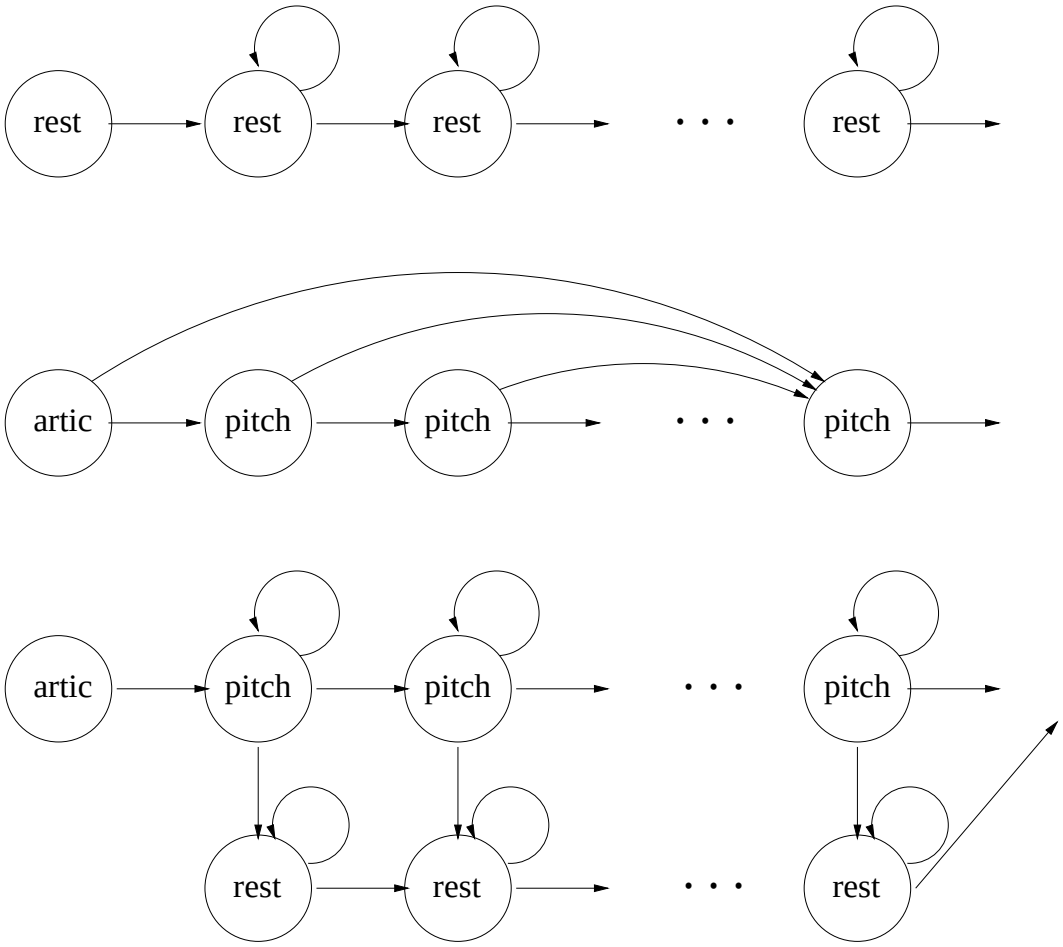

Figure 3: **Top:** A long rest model. **Middle:** A short note model. **Bottom:** A note model with optional rest at end. This model is used when a note is not required to sound until the beginning of the next note.

The parameters of these models can be either fixed *a priori* or trained from actual data if we have several different performances of the same piece. Parameters for the model in Figure 2 (bot), are estimated by the method of moments. That is, we choose n and p so that the mean, $n/(1-p)$, and the variance, $np/(1-p)^2$, approximate the empirical mean and variance for each note. Similarly, for the model in Figure 2 (top) we chose the transition probabilities to model a smoothed version of the empirical note length distribution. In the experiments presented in Section 7, we did not have enough data to train the length distributions, so they were chosen to be relatively uninformative distributions, however, in a more realistic setting we do train these distributions from actual data.

These are the basic ideas we use in practice, although numerous variations are possible. In addition, to completely specify the models, the states must be labeled and we assume that the output distribution depends only on this label and not the state itself (see Section 4). Figure 3 shows several examples of labeled note models. Observe that each of the note models has a unique state associated with the inception of the note (the first state of the model) and this state will be visited exactly once. We denote by start(m) the frame number (a random variable) corresponding to the inception of the m^{th} note.

The note models are then connected together in a left-right fashion with the final state(s) of each

model being connected to the initial state of the next model. We then append a rest with highly variable distribution to the beginning and end of the model to account for silence both before and after the performance.

4 The Data Model

In principle, every state, $\sigma \in \Sigma$, in our model has an associated output distribution, $P(Y = y|X = \sigma)$ describing the data we observe while visiting σ. The number of states in our model can reach the thousands, while the dimension of the frame vector, y, is also high — 256 in our experiments. Clearly simplifying assumptions are necessary if one hopes to train the $\{P(Y = y|X = \sigma)\}$ using a limited supply of data. We introduce three assumptions here that greatly reduce the number of parameters necessary to represent the data model. A more complete description of our data model is given in Section 6.

The three assumptions are as follows:

- Every state, $\sigma \in \Sigma$, is assigned a label, $l \in \Lambda$, where Λ is described in Eqn. 3. For each σ, the output distribution $P(Y = y|X = \sigma)$ is assumed to depend only on the label of σ. In other words, the output distributions of the states $\{\sigma \in \Sigma : L(\sigma) = l\}$ are "tied." This avoids pointless duplication in the training of output distributions.

- We compute a feature vector $s(y)$ assumed to contain all relevant information for inferences about the underlying state label. We assume that $P(Y = y|X = \sigma)$ depends only on the feature vector $s(y)$ and not on the entire data vector, y. This reduces the dimension of our observations to the dimension of $s(y)$.

- We partition $s(y)$ into a number of low-dimensional vectors which are assumed to be conditionally independent, given $X = \sigma$. This further reduces the dimension of the observations for which we must train output distributions.

It is difficult in practice to specify the output distributions *a priori*. Even when we can produce a reasonable first guess, we continue to hope that a training algorithm might improve upon it. An automatic means of training the output distributions makes our approach adaptable to changes in the setup such as a different solo instrument, room, placement of the microphone, or background "noise" from an accompanying instrument. We have used the well-known Baum-Welch or Forward-Backward algorithm to accomplish this task. This is described briefly in Section 6, however a more detailed description can be found in [13] or [14].

5 Segmenting the Data

Having trained the output distributions, we now attempt to segment our data. We define a *segmentation* as a vector $\mathbf{k} = (k_1, k_2, \ldots, k_M)$ giving an estimate of the starting times of the M notes (in frames). We discuss in this section several schemes for defining and computing the optimal segmentation.

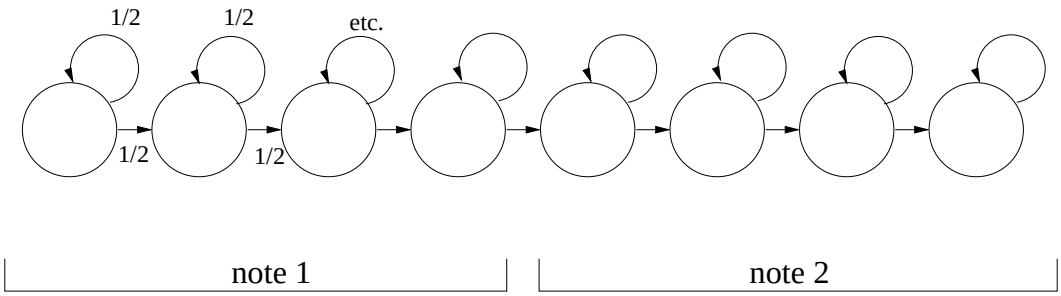

Figure 4: In the above situation, the most likely *a posteriori* state sequence is indifferent to the lengths of the two notes. This is undesirable since the prior model favors notes of equal length.

As already noted, by fixing $\mathbf{Y} = \mathbf{y}$ we induce a posterior distribution on $\mathbf{X}$, $P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y})$. One perfectly natural estimate of $\mathbf{X}$ would be the posterior mode

$$\hat{\mathbf{x}}_{\text{mode}} = \arg\max_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y})$$

This posterior mode is easily computed through dynamic programming [13]. Once we know the optimal state sequence $\mathbf{x}$, we could choose $\mathbf{k}(\hat{\mathbf{x}}_{\text{mode}})$ as our optimal segmentation where $\mathbf{k}(\mathbf{x}) = (k_1(\mathbf{x}), k_2(\mathbf{x}), \ldots, k_M(\mathbf{x}))$ has components defined by

$$k_m(\mathbf{x}) = \text{the unique } k \text{ for which } x_k = \text{start}(m)$$

The estimator $\mathbf{k}(\hat{\mathbf{x}}_{\text{mode}})$ produced reliable segmentations in many of our experiments so our criticism of it is based partly on philosophical grounds, however, we have observed situations where its performance was clearly suboptimal. A typical example is as follows.

Consider, for instance, the two-note model of Figure 4. To be specific, suppose that in Figure 4 the transition probabilities are all equal to $1/2$ (favoring notes of equal length) and that we have a total of 16 frames accounted for by this model. When the data is completely uninformative, (say both notes are the same pitch yet there is no clear rearticulation), the posterior distribution on X gives equal probability to every path. Thus $\mathbf{k}(\hat{\mathbf{x}}_{\text{mode}})$ has no preference for an 8-8 split over a 4-12 split. Clearly this is undesirable.

One solution to this problem would be to maximize the posterior likelihood on *segmentations* rather than paths:

$$P(\mathbf{K} = \mathbf{k}|\mathbf{Y} = \mathbf{y}) = \sum_{\mathbf{x}:\mathbf{k}(\mathbf{x})=\mathbf{k}} P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y})$$

but the new criterion still has weaknesses. Suppose we have two *nearly* identical segmentations with comparatively high posterior probability. If there is a single substantially different segmentation with slightly higher posterior probability, then the posterior most likely segmentation is the latter one. In our view, this choice is in error since a coarser discretization of the problem would collect the two nearly identical segmentations into a single segmentation with twice the posterior probability. In short, the posterior mode has the undesirable property of depending heavily on the choice of discretization. We do encounter this problem in practice and this leads us to question the validity of the posterior mode for our problem. We now turn to a different notion of optimality.

Let $\mathbf{k} = (k_1, k_2, \ldots, k_M)$ be an alleged segmentation of the data and suppose that $\mathbf{x}$ is the true state sequence. Let $H(\mathbf{k}, \mathbf{x})$ denote the number of segmentation errors defined by

$$H(\mathbf{k}, \mathbf{x}) = \#\{m : |k_m - k_m(\mathbf{x})| > \eta\}. \tag{5}$$

Here η is a threshold describing the smallest error that we are willing to call a segmentation mistake. (In our experiments we have used a value of $\eta = 1$ or 32 ms.) We define our segmentation estimate

$$\hat{\mathbf{k}} = \arg\min_{\mathbf{k}} E[H(\mathbf{k}, \mathbf{X})|\mathbf{Y} = \mathbf{y}].$$

Owing to the easy computability of the posterior distribution on $\mathbf{x}$, we can in fact find $\hat{\mathbf{k}}$. Observe

$$\begin{aligned} E[H(\mathbf{k}, \mathbf{X})|\mathbf{Y} = \mathbf{y}] &= E[\sum_{m=1}^{M} 1(|k_m - k_m(\mathbf{X})| > \eta)|\mathbf{Y} = \mathbf{y}] \\ &= \sum_{m=1}^{M} P(|k_m - k_m(\mathbf{X})| > \eta|\mathbf{Y} = \mathbf{y}). \end{aligned} \tag{6}$$

We can minimize this quantity by minimizing each of the terms in Eqn. 6, or by maximizing the terms

$$P(|k_m - k_m(\mathbf{X})| \leq \eta|\mathbf{Y} = \mathbf{y}) = \sum_{k=k_m-\eta}^{k_m+\eta} P(X_k = \text{start}(m)|\mathbf{Y} = \mathbf{y})$$

For this last quantity, we compute the posterior marginal distributions, $P(X_k = \sigma|\mathbf{Y} = \mathbf{y})$, through the forward-backward equations described in Section 6. This is the method we have used in the experiments of Section 7.

The ideas of this section can be adapted easily to the "on-line" segmentation problem used for real-time accompaniment. In this problem we wish to know when each note onset has occurred, and we would like this information shortly after the onset has occurred. We accomplish this in two stages. Suppose we have detected all notes before note m and are currently waiting to locate note m. We examine successive frames, k, until

$$P(X_k \geq \text{start}(m)|Y_1 = y_1, Y_2 = y_2, \ldots, Y_k = y_k) > \tau \tag{7}$$

for some threshold τ. Note that the left-right nature of the model means the states are ordered so it makes sense to say that $\sigma_1 \geq \sigma_2$ for $\sigma_1, \sigma_2 \in \Sigma$. The left side of Eqn. 7 is accomplished through the recursive computation of the α-probabilities (see Section 6) which, after normalization, give $P(X_k = \sigma|Y_1 = y_1, Y_2 = y_2, \ldots, Y_k = y_k)$.

When Eqn. 7 first holds, say for frame k^*, we assume that the mth note onset is in the past. At this point we compute our estimate, $\hat{k}_m$, of the mth note onset by

$$\hat{k}_m = \arg\max_{k \leq k^*} P(X_k = \text{start}(m)|Y_1 = y_1, Y_2 = y_2, \ldots, Y_{k^*} = y_{k^*})$$

We use the forward-backward algorithm to compute $P(X_k = \sigma|Y_1 = y_1, Y_2 = y_2, \ldots, Y_{k^*} = y_{k^*})$. With reasonably chosen threshold, τ, we find only modest differences between the on-line estimates and the off-line estimates, however, the more difficult note detections, such as rearticulations, are made with greater time lag than the easier ones. We have not performed any systematic experiments with τ, however, clearly τ represents a tradeoff between latency and accuracy.

6 Representing and Training the Output Distributions

This section addresses the representation and training of the output distributions $\{P(Y_k = y|X_k = \sigma)\}$. For simplicity of notation we refer to these distributions as $\{p(y|\sigma)\}$ in this section since the output distributions do not vary with the frame, k.

6.1 Representing the Output Distributions

As described in the Section 4, each state $\sigma \in \Sigma$ is given a label, $l = L(\sigma) \in \Lambda$, characterizing what we expect to hear when $X_k = \sigma$. The possible labels are enumerated in Eqn. 3. This assumption is made precise by the statement

$$p(y|\sigma) = p(y|L(\sigma)).$$

for some probability functions $\{p(y|l)\}$, $l \in \Lambda$. This assumption limits the number of output distributions we must represent and eventually train to $|\Lambda|$.

A frame of data, y, lies in a high dimensional space — $\Re^J$ where $J = 256$ in our experiments. In order to make our representation and training possible, we must also reduce the dimensionality of the y-space and we do this by using the notion of sufficiency. Let s be a vector-valued function mapping $\Re^J$ into a much lower dimensional space. We think of the statistic, $s(y)$, as containing all relevant information for estimating the label, $l \in \Lambda$, given no information other than y. That is, we assume s is a *sufficient statistic* for l, meaning that $p(y|s(y), l)$ does not depend on l. A simple consequence is that

$$\begin{aligned} p(y|l) &= p(y, s(y)|l) \\ &= p(y|s(y), l)p(s(y)|l) \\ &= p(y|s(y))p(s(y)|l) \end{aligned}$$

Since our frame data is fixed, $p(y|s(y))$ is a constant for each frame and we are concerned only with the way $p(y|l)$ depends on l so we disregard the $p(y|s(y))$ factor and concentrate on representing and training the $p(s|l)$ factor.

Although we have significantly reduced the complexity of our data model, we still make two further simplifications. First we assume that our sufficient statistic, $s(y)$, can be written

$$s(y) = (s_1(y), s_2(y), \ldots, s_D(y))$$

where the vector-valued components $\{s_d\}$ are conditionally independent given the state label. For each statistic, s_d, $d = 1, 2, \ldots, D$, we define a "signature" function

$$r_d : \Lambda \to \{0, 1, \ldots, V-1\}.$$

We call the vector of these functions, $r(l) = (r_1(l), r_2(l), \ldots, r_D(l))$, the *signature* of $l \in \Lambda$. The idea is that if we anticipate that two labels have the same distribution for the d^{th} output component, s_d, then their modeled output distributions should be the same for the s_d component and they should share data in training these output distributions. That is, we assume $p(s_d|l) = p(s_d|r_d(l))$. In summary,

$$\begin{aligned} p(y|\sigma) &= p(y|l) && [l = L(\sigma) \text{ is label}] \\ &\propto p(s|l) && [s = s(y) \text{ is sufficient statistic}] \\ &= \textstyle\prod_{d=1}^{D} p(s_d|l) && [\text{the } \{s_d\} \text{ are conditionally independent}] \\ &= \textstyle\prod_{d=1}^{D} p(s_d|r_d(l)) && [r_d = r_d(l) \text{ is signature}] \end{aligned} \tag{8}$$

A more specific look at our data model will illuminate this discussion. Our label set, Λ, can be partitioned into two classes — those for which the conditional distribution on a frame of data y, $p(y|l)$, favors low amplitude signals, {rest}, and those for which $p(y|l)$ favors relatively higher amplitude signals, $\Lambda \setminus \{\text{rest}\}$. Thus we define a statistic to measure the average amplitude of a frame of data

$$s_1(y) = \sum_{j=0}^{J-1} y^2(j)$$

and define a signature function r_1 that respects the above partition of Λ — that is, for example, $r_1(\text{rest}) = 0$, $r_1(\text{not rest}) = 1$. Note that, since r_1 takes only two values, there are only two one-dimensional distributions, $p(s_1|r_1 = 0), p(s_1|r_1 = 1)$, needing training. If every frame in a data set were labeled with its true $l \in \Lambda$, then every $s_1(y_k)$, $k = 0, 1, \ldots, K-1$ would be an example from one of these two distributions. Thus, only a modestly-sized data set is necessary for training the two distributions. If one accepts the assumptions of our model, most importantly the conditional independence of the $\{s_d\}$, then, as statistics are added to the model there is no decrease in the data available for the training of each conditional distribution — the statistics do not compete with one another for training data.

Similarly, define the statistic

$$s_2(y) = \left(\sum_{j=1}^{\lfloor J/3 \rfloor} y^2(j) - 2 \sum_{\lfloor J/3 \rfloor + 1}^{\lfloor 2J/3 \rfloor} y^2(j) + \sum_{\lfloor 2J/3 \rfloor + 1}^{J-1} y^2(j) \right) / s_1(y)$$

which measures the local "activity" of the signal. Figure 5 shows the way this statistic behaves on the data of Figure 7 (top). This statistic partitions Λ into those labels for which we expect much activity, $\{\text{artic}(1), \text{artic}(2), \ldots, \text{artic}(P)\}$, those for which we expect little activity, $\{\text{pitch}(1), \text{pitch}(2), \ldots, \text{pitch}(P)\}$, and, just to be on the safe side, those for which we have no clear expectation, {rest}. (We might have been able to include the rest in the latter category were it not for the danger imposed by the normalization). We define r_2 to respect this partition, thus creating 3 more output distributions needing training. Notice that the total number of distributions is only the *sum* of the partition sizes for the $r_1, r_2, \ldots, r_D$.

The rest of the statistics are designed to discriminate among the various pitches, so they are based on the frequency content of the signal. Let $\tilde{y}$ be the squared modulus of the finite Fourier transform of y having values for $\omega = 0, 1, \ldots, J/2 - 1$. We divide the frequency axis into a number of disjoint intervals, I_d, indexed by $d = 3, 4, \ldots, D$. For each of these intervals we compute a vector-valued statistic $s_d(y)$ defined by

$$s_d = (\arg\max_{\omega \in I_d} \tilde{y}(w), \text{var}(I_d)) \tag{9}$$

where $\text{var}(I_d)$ is the variance of $\tilde{y}$ suitably normalized to be a probability distribution on the interval I_d. If the variance has a relatively low value, it is likely that a harmonic is present in I_d. In this case the first component estimates the frequency of that harmonic. When the variance component has a relatively high value, it is likely that no harmonic exits in I_d. Pairs of notes corresponding to simple ratios of frequencies (octaves, fifths, etc.) will share certain harmonics. For these pitch discriminating statistics, r_d is chosen to be constant among notes that (approximately) share harmonics in the interval I_d.

As already mentioned, if our modeling assumptions hold then adding statistics to our model produces no decrease in the training data available for estimating their conditional distributions. As we

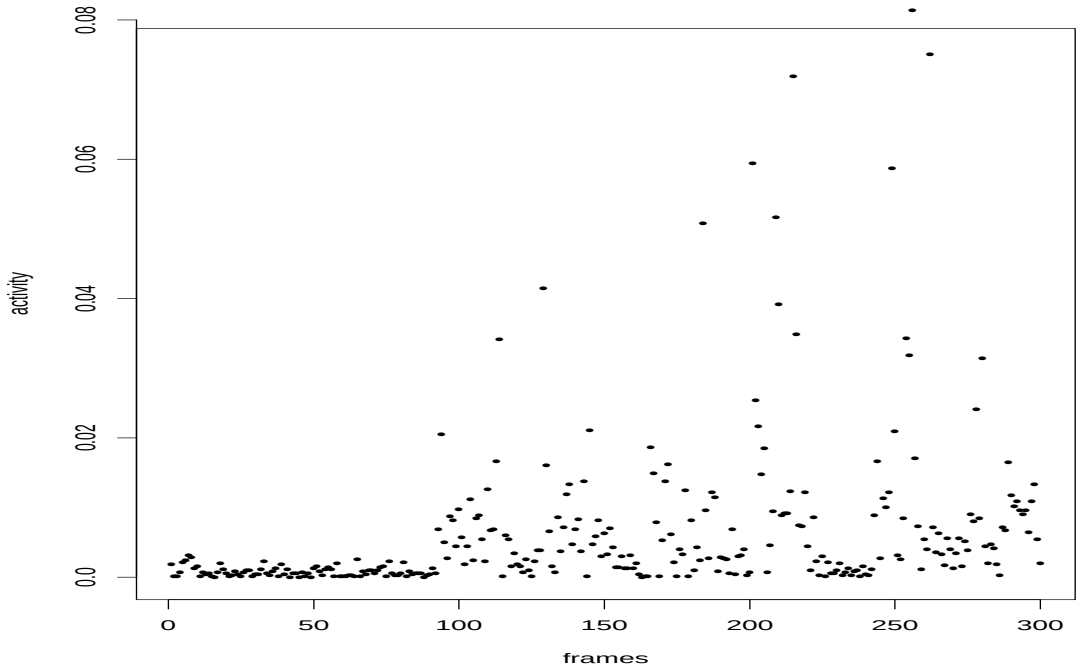

Figure 5: The statistic s_2 computed for the data in Figure 7 (top). The beginnings of notes as well as the rearticulation cause comparatively large values for the statistic.

increase the number of statistics, however, it becomes more difficult to maintain the plausibility of conditional independence of the $\{s_d\}$ given l. Violations of this assumption over-emphasize the importance of some of the factors in Eqn. 8; (Imagine the effect of including two *identical* statistics). Thus our paradoxical "free lunch" is actually paid for by our willingness to accept a model that becomes progressively less believable as more and more statistics are added. We try to minimize conditional dependencies among the $\{s_d : d = 3, 2, \ldots, D\}$ by having the statistics depend on distinct regions of the frequency spectrum, but we should expect that the conditional independence assumption is only approximate.

Finally, the statistics are discretized. The original discretization is chosen so that the unconditional empirical distributions of the statistics are uniform. Later on, however, during the training phase, we redefine the discretizations through a recursive splitting procedure that seeks to maximize the mutual information between the quantized statistics and the estimated label signatures. To minimize the complexity of notation, the $\{s_d\}$ refer to the *quantized* statistics in what follows.

6.2 Training the Output Distributions

Once we choose the score and fix the transition probabilities through training or informed assignment, our prior model for $\mathbf{X}$ is determined and this remains fixed throughout the Forward-Backward algorithm — we do not use the algorithm to reestimate transition probabilities as is usual. We begin by initializing the output distributions, $\{p(s_d|r_d)\}$. The algorithm is not particularly sensitive to this initialization so we need not exercise much care in doing so. In practice, we have initialized the distributions near their anticipated values, however we do not know that this is necessary. With our probability model (X, Y) now in place, we can define the quantities

$$\begin{aligned} \alpha_k(x_k) &= p(x_k, y_0, y_1, \ldots, y_k) \\ \beta_k(x_k) &= p(y_{k+1}, y_{k+2}, \ldots, y_{K-1}|x_k) \end{aligned}$$

$k = 0, 1, \ldots, K-1$, known respectively as the forward and backward probabilities. The $\{\alpha_k\}$ and $\{\beta_k\}$ satisfy the recursions

$$\alpha_{k+1}(x_{k+1}) = \sum_{x_k} \alpha_k(x_k) p(x_{k+1}|x_k) p(y_{k+1}|x_{k+1}) \tag{10}$$

$k = 0, 1, \ldots, K-2$, and

$$\beta_{k-1}(x_{k-1}) = \sum_{x_k} \beta_k(x_k) p(x_k|x_{k-1}) p(y_k|x_k)$$

$k = K-1, K-2, \ldots, 1$. We can initialize these recursions by letting $\alpha_{-1}(\cdot)$ and $\beta_K(\cdot)$ concentrate unit masses on the initial and final states of the model.

From the forward and backward probabilities, one easily computes the posterior marginal distributions:

$$P(X_k = x_k|\mathbf{Y} = \mathbf{y}) = p(x_k|\mathbf{y}) = \frac{\alpha_k(x_k)\beta_k(x_k)}{\sum_{x'_k} \alpha_k(x'_k)\beta_k(x'_k)} \tag{11}$$

To understand how we might make use of Eqn. 11, imagine that one of the $p(x_k|\mathbf{y})$, viewed as a function of x_k, concentrates all its probability on a set of states all having label $l \in \Lambda$. Then we would be safe to use y_k as an example of the distribution $p(y|l)$. Enough samples of this kind would make

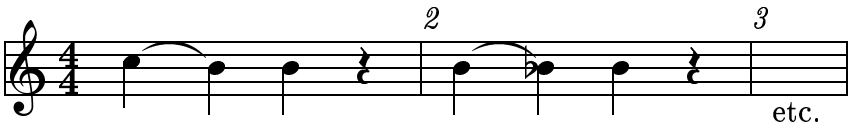

Figure 6: The above shows the first few measures of our training score. This musical excerpt is ideal for training since it is easy to segment and contains many examples of each of the statistics we wish to train.

a supervised learning algorithm feasible. The forward-backward algorithm is a variation on this idea. After having computed the $\{p(x_k|\mathbf{y})\}$, we re-estimate the distributions $p(s_d|r_d)$ by the formula

$$p(s_d = q|r_d = i) = \frac{C_d(q,i)}{\sum_{q'} C_d(q',i)}$$

where

$$C_d(q,i) = \sum_{\{k:s_d(y_k)=q\}} p(r_d(X_k) = i|\mathbf{y})$$

We are now free to again re-estimate the $\{p(s_d|r_d)\}$ using our new $\{p(s_d|r_d)\}$ in the calculations of Eqn. 11. The sequence of parameter estimates for the $\{p(s_d|r_d)\}$ is guaranteed to converge to a local optimum of the data likelihood [14].

7 Experiments

Before we can demonstrate our segmentation algorithm on a real example, we must first train the output distributions using the methods of Section 6.2. We do this using a very simple score, purely for illustrative purposes. We have chosen the musical excerpt shown in Figure 6, which provides plenty of examples from each of the output distributions we wish to train.

Due to the large number of states in our graph Σ, it is not possible to carry through the training algorithm precisely as described in Section 6.2; in practice, we must modify the algorithm to accommodate pruning. To approximate Eqn. 11, we first perform a forward pass through the data in which we compute the forward probabilities of Eqn. 10. This computation is approximate since, after each iteration of Eqn. 10, we eliminate states whose (normalized) forward probabilities fall below a certain threshold. Then in the backward pass, in each iteration we only compute the backward probabilities for states having non-zero forward probabilities. The effects of this approximation are negligible in our experiments.

The top panel of Figure 7 shows the first few seconds of the data used in training the pitch oriented distributions — a spectrogram. The bottom two panels show the $P(Y = y|X = x)$ distributions for several locally relevant labels before and after 5 iterations of the training algorithm. This figure demonstrates both the weakness of our initial guess and the robustness of the training algorithm to this weakness.

After training we can demonstrate our segmentation algorithm on the much more challenging score of Figure 8 — a cadenza to the first movement of the Mozart Oboe Concerto, written and performed by

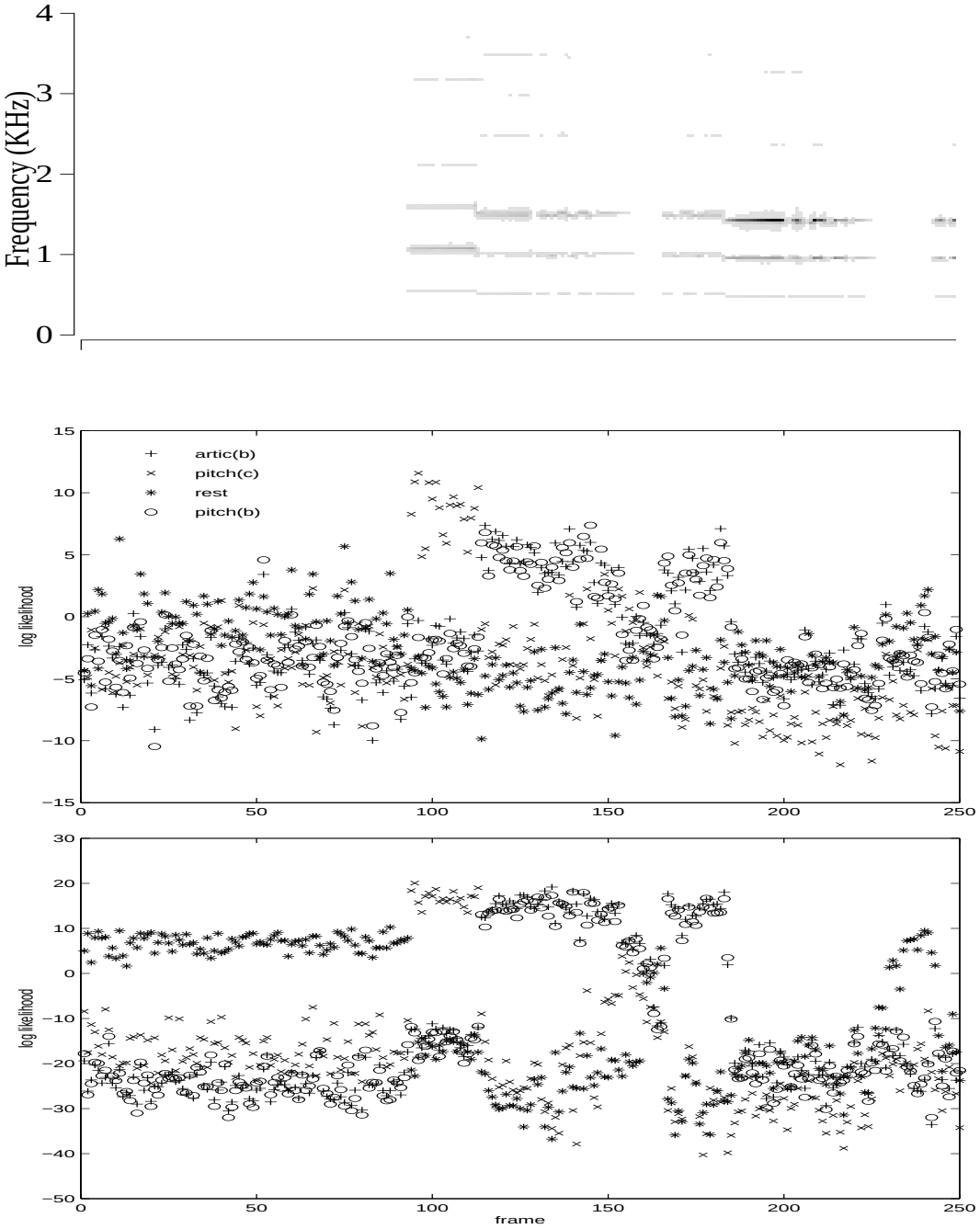

Figure 7: **Top:** A spectrogram of the training data, (compare with musical excerpt). **Middle:** The log likelihoods of 4 different labels before training. **Bottom:** The same log likelihoods after 5 iterations of the training algorithm. 1 frame is 32 ms.

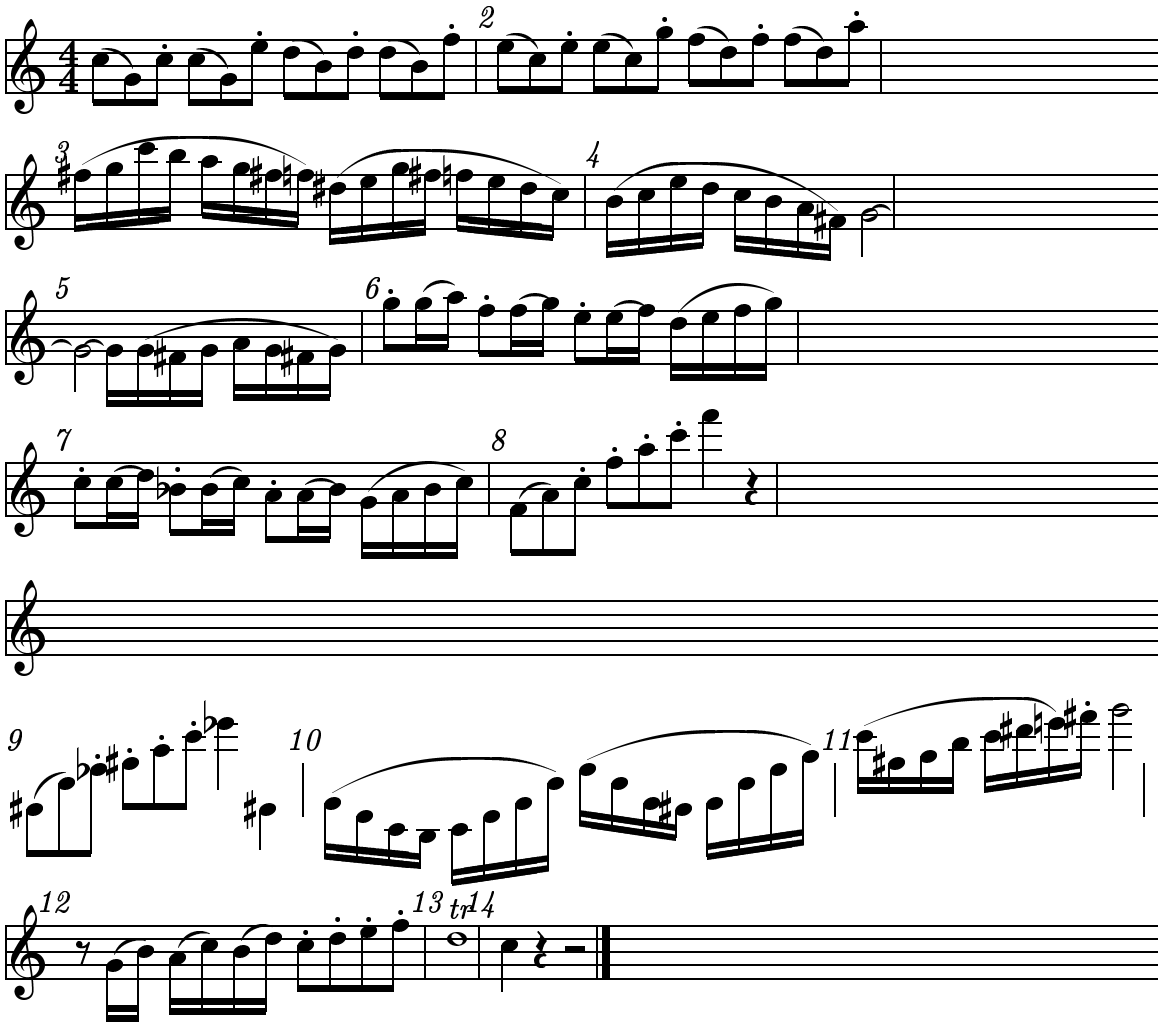

Figure 8: Cadenza to 1st Movement of Mozart's Concerto for Oboe and Orchestra K. 314

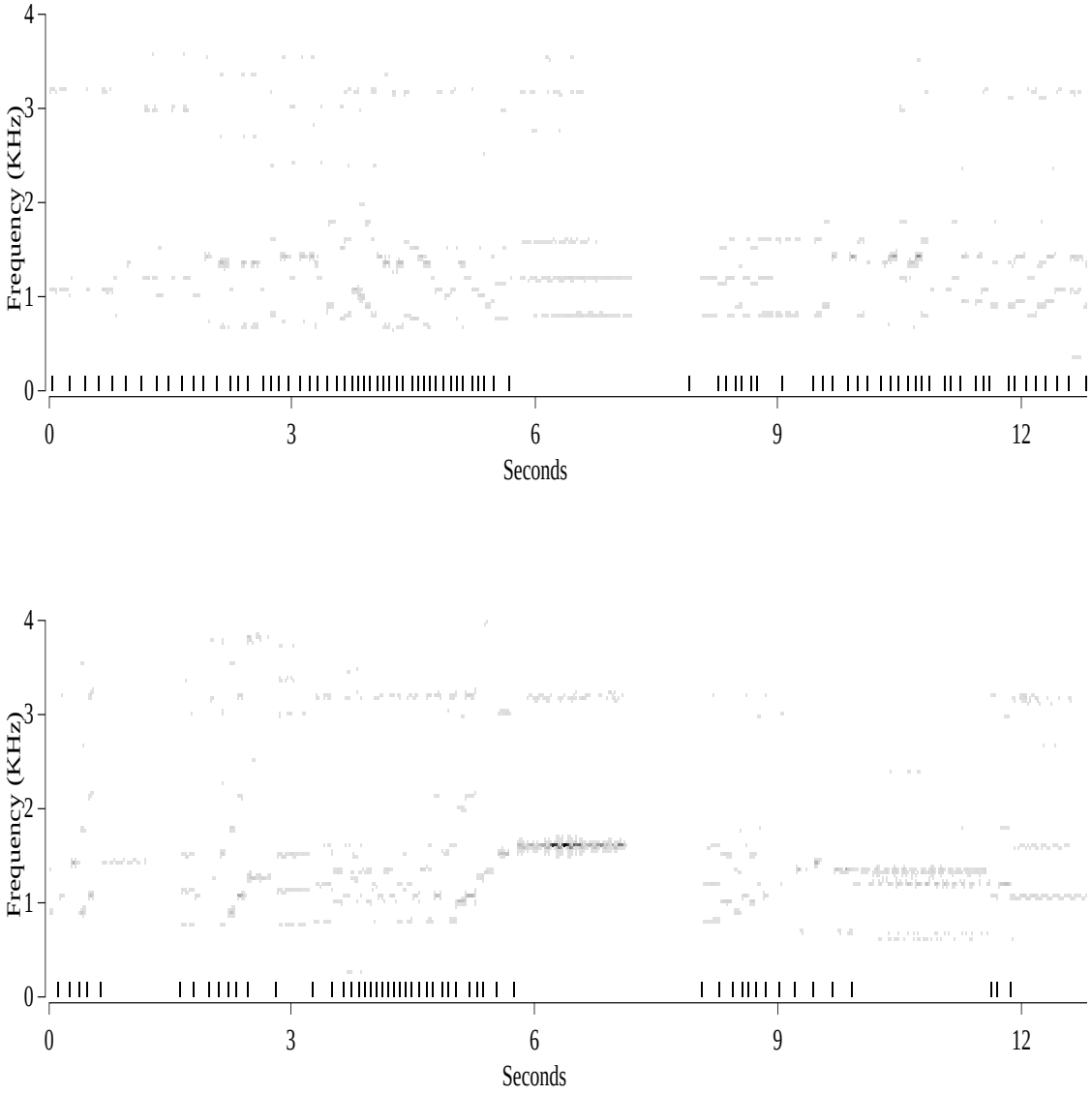

Figure 9: The top and bottom figures are the 1st and 2nd halves of the spectrogram of the cadenza data. The tick marks along the bottom show the estimated beginnings of the notes.

the author. The example was chosen for the difficulty imposed by the extreme variability of tempo in a cadenza. In minimizing the posterior expected number of segmentation errors, we say a segmentation error occurs when an estimated note onset misses the true location by more than one frame ($\eta = 1$ in Eqn. 5 — this is an error of 32 ms. in our experiments). Figure 9 shows the spectrogram data for the cadenza performance with the optimal segmentation superimposed. Unfortunately, it is only possible to determine from the figure that the segmentation is more or less correct and this is true of any visual representation. In our opinion, however, the results are really much better than this. We have produced aural "pictures" of our segmentations by superimposing clicks at the alleged beginnings of each note and listening to the results. The average listener finds essentially no fault in most of our segmentations judging by these click-files. More careful inspection reveals some inaccuracies due to the discretization into frames and these are most obvious in series of rapid notes (say sixteenth notes at a quarter note = 120 bpm), and the occasional misplacement of a rearticulated note — an event also sometimes difficult for the person to locate. We encourage the reader not to take our word for this, but rather to reach an independent conclusion. For this purpose we have the above cadenza example available at http://fafner.math.umass.edu/parsing. Examples of our current state-of-the art in automatic accompaniment can be heard at http://fafner.math.umass.edu/music_plus_one.

8 Future Work

We believe that the approach presented here can, in principle, be extended to small collections of instruments. In doing so, we view the score as a sequence of chords rather than a sequence of notes. Any change of state of any of the instruments would result in a new chord, even if the remaining instruments sustain their pitches. In our model, each state is now associated with a chord and one can compute the spectral distribution of a chord by examining the contributions from each constituent note. Thus, given that a particular chord is sounding, we have an expectation of what we will observe in each of the window-based pitch statistics defined in Eqn. 9. The chord probability distribution can be built out of these statistics just as we built the single note distributions, making the same conditional independence assumption.

There are several possible pitfalls with this approach that should be mentioned. First, certain chords are virtually indistinguishable through their spectral distributions. For example, the spectral distribution of single note does not change appreciably by adding the octave or twelfth above it since no new harmonics are introduced. Our proposed method would have difficulty in determining boundaries that involve making this distinction — a very unusual case. Second, this method assumes that the various instruments play in perfect synchronicity so chord changes happen exactly as indicated in the score with no intermediate chords being introduced due to inaccuracies. This assumption is, of course, only an approximation and a deterioration in performance might result. However, when parts do not move in synchronicity it is not clear what the "right" answer is when identifying the new chord onset, so this weakness might be minor.

References

[1] B. Vercoe, M. Puckette "Synthetic Rehearsal: Training the Synthetic Performer," *Proceedings of the International Computer Music Conference, 1985* 275–278, Burnaby, British Columbia, Canada,

1985.

[2] B. Vercoe, "The Synthetic Performer in the Context of Live Performance," *Proceedings of the International Computer Music Conference, 1984* 199–200, IRCAM Paris, France, 1984.

[3] R. Dannenberg , "An On-Line Algorithm for Real-Time Accompaniment," *Proceedings of the International Computer Music Conference, 1984* IRCAM Paris, France, 193–198, 1984.

[4] J. Bloch, R. Dannenberg, "Real-Time Computer Accompaniment of Keyboard Performances," *Proceedings of the International Computer Music Conference, 1985* 279–289, Burnaby, British Columbia, Canada, 1985.

[5] R. Dannenberg, H. Mukaino "New Techniques for Enhanced Quality of Computer Accompaniment" *Proceedings of the International Computer Music Conference, 1988* 243–249, Köln, 1988.

[6] B. Baird, D. Blevins, N. Zahler, "Artificial Intelligence and Music: Implementing an Interactive Computer Performer" *Computer Music Journal* 17:2, 73–79, 1993.

[7] J. Brown, "Musical Fundamental Frequency Tracking Using a Pattern Recognition Method," *Journal of the Acoustical Society of America* 92:3, 1992, 1394–1402.

[8] L. Grubb, R. Dannenberg, "A Stochastic Method of Tracking a Vocal Performer," *Proceedings of the International Computer Music Conference, 1998* 301–308, 1997.

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

[10] K. F. Lee, "Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The Sphinx System," Ph.D. Thesis, Computer Science Dept. Carnegie Mellon Univ. Pittsburgh, PA, 1988.

[11] G. Kopec and P. Chou, "Document Image Decoding Using Markov Source Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 16, 602–617.

[12] T. Imai, R. Schwartz, F. Kubala, and L. Nguyen. "Improved Topic Discrimination of Broadcast News Using a Model of Multiple Simultaneous Topics," Proc. ICASSP-97, 727–730, April, 1997.

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

[14] L. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes," *Inequalities III* New York, Academic Press, Edited by: Oved Shisha, 1–8, 1967.

Biography

Christopher Raphael was born in Hayward, CA in 1960. He received his MS in Computer and Information Science from the University of CA at Santa Cruz in 1984 and his Ph.D. in Applied Mathematics from Brown University in 1991. He has been an NSF Postdoctoral Research fellowship and is currently on the faculty in the Mathematics and Statistics Department at the University of Massachusetts at Amherst. As a winner of the San Francisco Symphony Young Artist Competition, he soloed with that orchestra in 1978 and has held a fellowship at Tanglewood.