CS379H Undergraduate Thesis

Polyphonic Guitar Transcription by Bayesian Methods

Mark Kedzierski
University of Texas at Austin,
Department of Computer Sciences
2005

Abstract

I present a method of transcribing Polyphonic guitar music using probabilistic modeling. The generative model an expanded version of the one found in (Cemgil, 2004). The piano roll, or score, of the given audio waveform is inferred using MAP (Maximum A Posteriori) trajectory estimation via a Switching Kalman Filter. Because exact inference is generally intractable on Switching Kalman Filters, pruning and approximation methods are used. Integral approximation is achieved by Kalman filtering recursions.

To make the model useful, and to expand on previous work, (Kedzierski, 2004), my generative model applies many restrictions to limit the size of the state space. The performance is restricted to a single guitar which can play at most 6 simultaneous notes. The Prior probability over the notes is novel and includes knowledge of tonality, tempo, composition structure, as well as the physical distances between notes on the neck of the guitar.

The goal of this paper is to step closer to a real-time polyphonic MIDI sequencer Virtual Studio Technology plug-in. This could be used for composition, performance, and the driving of virtual musicians/instruments.

Table of Contents

Introduction

In this report I will outline a solution to the problem of polyphonic music transcription based on Bayesian inference. Specifically, transcribing a digital recording of a single guitar player. The applications of this capability include music education, composition, and recording.

Transcription is strongly related to the measurement of pitch in an audio signal; a problem which has been long investigated due to it's many potential appications. It is a difficult one, however, because pitch is a subjective attribute which cannot directly be measured. Pitch is defined as the characteristic of a sound which places it at a specific location in the musical scale. The ability to detect it varies among different people. Some are born with "perfect pitch", which means that they can accurately classify the pitch of a sample sound without any reference. Most people however, can only detect intervals, or the difference in pitch between different sounds. Some suggest that pitch detection might be a learned ability, as different cultures often have different musical scales, each with different intervals. In any case, it is difficult to detect any attribute which is, by definition, subjective. The first approach I took to solving this problem was to determine the fundamental frequency of the audio sample, in blocks (windows), and use it to classify its pitch. While this approach resulted in an effective real-time algorithm to detect the pitch of monophonic digital audio, it is ineffective in polyphonic applications.

In this paper I explore a method of transcribing Polyphonic guitar music using probabilistic modeling. First, we define a state space model, based on (Cemgil, 2004), for generating audio signals not unlike those of stringed instruments. We then use our generative model to infer the most likely *piano roll,* i.e. a musical score, which would generate the audio waveform. The piano roll describes the musical performance which caused the digital audio data. This is done by generating waveforms for every possible piano roll and comparing them to the source. The closest waveform would give us the correct piano roll. In practice, (because we can't realistically make all those calculations), we do this in real-time as a Viterbi MAP state trajectory estimation problem using Switching Kalman Filters (Murphy, 1998). This means we propagate the most likely sequence of states which led to the current audio sample.

The implementation of my research has been developed in Matlab. The program is capable of learning parameters from sampled audio data as well as Viterbi MAP trajectory estimation for polyphonic music. I planned to implement the code as a VST plug-in, but didn't have time. VST would allow the product to be used in conjunction with a variety of professional audio software.

The remainder of this document consists of several sections. I have attempted to order the document in such a way that all necessary background theory is presented before its application. However, some background in probability/statistics is assumed. In the first section I mention related works. Next I describe those physical properties of sound waves which are relevant to music. Third, I provide a primer in the music theory necessary to understanding the implementation. Fourth, I explain the MIDI standard and how it enables digital instruments to communicate. Finally, I outline the solution of a simplified problem, as well as the algorithms/models utilized in the final implementation. To close the document I summarize results from actual experiments and propose future enhancements the system.

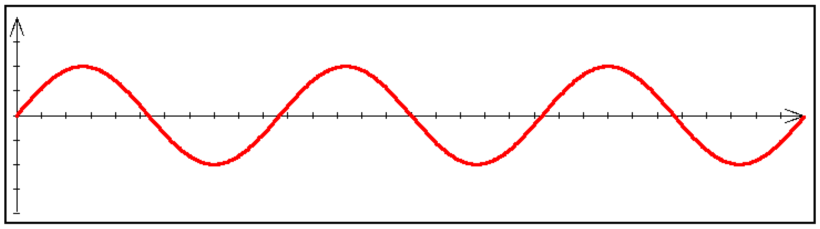
Related Works

This is work is largely based on a Ph.D. thesis by Ali Cemgil (Cemgil, 2004). He outlined a generative Switching Kalman Filter model for plucked string audio waveforms as well as Bayesian inference algorithms to implement polyphonic music transcription. The model is defined, in general, as a Dynamic Bayesian Network (DBN) such as those defined in (Murphy, 2004). The solution works very effectively for monophonic melodies and usually works for polyphonic. The errors in polyphonic transcription case arise from the hill-climing algorithm getting stuck on local maxima.

In this paper I propose to increase performance of inference by expanding the prior distribution over piano rolls with knowledge of tonality and by applying various limitations to Cemgil's generative model. I also describe my implementation of the inference and learning algorithms using Matlab.

Physical Properties of Sound Waves

In this section I discuss the physical nature of sound as relevant in the discussion of musical tones and/or melodies. We define pitch as any the attribute of sound which defines its placement on the musical scale. We define a tone as any sound which is associated with a pitch. First I offer a concise explanation by Brian C.J. Moore: “Sound originates from the motion or vibration of an object. This motion is impressed upon the surrounding medium (usually air) as a pattern of changes in pressure. What actually happens is that the atmospheric particles, or molecules, are squeezed closer together than normal (called condensation) and then pulled farther apart than normal (called rarefaction).” (Moore, 1999) We obtain a graphical representation of a sound be plotting the amplitude of pressure variation over time. We call this plot a waveform. I provide an example below. Note because waveforms are a plot against time, they are said to exist in the time-domain.



This waveform is known as a pure-tone; plotted as pressure variation over time. Remember that a tone is defined as any sound which can be classified as having a pitch. A gunshot is not a tone. A pure tone is named for its very “clean” musical sound, very much similar to that of a tuning fork. A pure tone’s waveform is simply a sine wave, whose frequency determines where the note lies on the musical scale. For example, the commonly used A-440 tuning fork produces a wave very similar to a 440 Hz sine wave. We will discuss this in greater detail in the next section.

Figure 1

While pure tones are important in defining pitch and useful in theoretical discussion, we do not encounter them in nature. Instead, we see complex-tones. Like pure-tones, all complex tones are assigned a unique pitch value. But instead of consisting of a single sine wave at a given frequency, they contain several superimposed sine waves, all located at harmonics. More accurately, a complex tone consists of a fundamental frequency and several overtones.

We define the ith harmonic of a given frequency as the ith integral multiple of said frequency. For example, the 0th, 1st, and 2nd harmonics of a 440Hz tone would exist at 440Hz, 880Hz, and 1320Hz, respectively. We define the fundamental frequency of a complex tone to be the 0th harmonic. We similarly define any tones which occur at subsequent harmonics as overtones of that fundamental frequency. The next figure depicts sine waves of frequencies at 3rd, 4th, and 5th harmonics combined to form a complex tone.

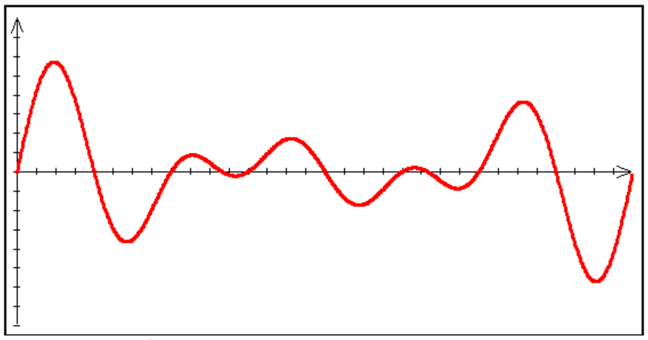

Complex tone, containing 3 overtones.

Figure 2

The sound generated from a plucked guitar string is an example of a complex tone. The waveform is shown below. Note how it resembles a damped sin wave, with the amplitude decreasing over time.

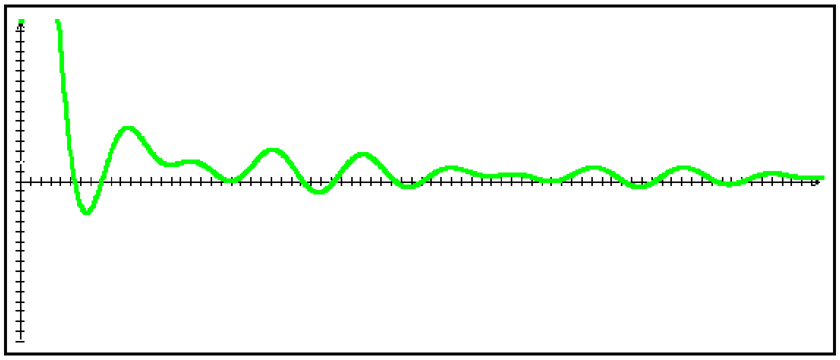

Sampled waveform of a plucked guitar string. This is an example of a complex tone.

Figure 3

Music Theory Primer

To begin our primer on music theory, we define a note as any named tone. The ordered collection of all possible notes is known as the musical scale. We define an interval as the spatial distance, on the musical scale, between successive notes. Commonly occurring intervals are given names. The smallest interval is the ½-step (or semitone), which is used to represent the distance between two adjacent notes on the scale. The Whole-step is the equivalent of two ½-steps. Some other commonly occurring intervals are the Fifth, Major Third and Minor Third. We can now define a melody as a series of notes played at varying intervals.

The musical scale is logically divided into groups of twelve notes known as octaves. Every octave consists of twelve semitones which are labeled in the following manner: C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#/Ab, A, A#/Bb, B. We use this scheme because each of these labels represents a unique "sound", only twelve of which exists. This means that a C note sounds the "same" as a C note in any other octave. Each octave in the musical scale is labeled by an index, starting at 0. In turn, every note on the musical scale can be uniquely identified by a note label and octave index. Examples are: C0, Db5, and G2. Middle C is represented as C4. So, if we know the pitch of a tone, we know where it belongs on the musical scale. But since real instruments don't make pure tones, we need a way to assign pitch. "Since pitch is a subjective attribute, it cannot be measured directly. Assigning a pitch value to a sound is generally understood to mean specifying the frequency of a pure tone having the same subjective pitch as a sound." (Moore, 1999). We define the relative frequency (ç) of any note ç as the frequency of said pure tone. For example, (C4) = 261.4Hz. Next we discuss how the musical scale maps to the human audible frequency range.

The human audible frequency range is approximately 20-20,000Hz. However, we are not concerned with anything above 5Khz, as we are unable to discern the pitch of frequencies in the that range. (Moore, 1999) The range of 0-5kHz is partitioned into eight octaves, each beginning at C, where (C0)=16.4Hz, (C1)=32.7Hz, (C8)=4186Hz. It may be clear now why C4 is known as middle C. You may notice that (C1) = 2(C0). This leads us to the real definition of the octave as the interval between two tones when their frequencies are in a ratio 2:1 (Moore, 1997) We can deduce from this that the eight octaves are not the same size, instead they increase exponentially. Every interval, in fact, is actually a relationship between two tones. The following table depicts common intervals and their associated ratios:

Interval	Ratio
½ step	1:1.05946
Fifth	3:2
Maj Third	5:4
Min Third	6:5

Description of common intervals

Table 1

MIDI Primer

The Musical Instrument Device Interface is widely used by musicians to communicate between digital musical instruments. It is a standard which has been agreed upon by major manufacturers in the industry. It connects not only musical instruments, but all devices such as recorders and lighting systems.

The communication in a MIDI system is done by sending and receiving messages. Examples of some messages related to melody composition are Note-On, Note-Off, Tempo Change, Key-Signature Change, and Time-Signature change. The meanings behind these events are self-explanatory. Each message contains variables specific to the message. For example, a Note-On event would contain information about which note should be played, and the volume at which it should be played. Each note is assigned a specific index. These indices are standard and are listed in Appendix A. The Key-Signature message would contain the new desired key signature.

In practice a synthesizer driver would create and dispatch a series of MIDI messages to a synthesizer. The synthesizer would then playback the desired notes using pre-recorded instrument audio samples.

State Space Models and Linear Dynamical Systems

Bayesian inference problems involve making estimates of the current *state* of a graphical *state-space* system model, given uncertain or incomplete data. This consists of defining a graphical system model and then performing inference algorithms on the model.

■ Dynamical *State-Space* Models

A *state-space* system model consists of a set of equations which synthesize the output of some physical system. Though not always the case, in this paper I limit discussion to systems which operate/evolve as a function of time (a.k.a. Dynamical Systems).

The system is defined, among other things, by the set of possible 'situations', or *states,* that it can be in at any particular time. The state variable, s_t, represents the current state of the system at each time t, {t: 0≤t≤T}. This can be represented as a single variable, or a vector; with continuous or discrete values. The length of a state vector, $| s_t |$, is labeled as N (N=1 in single variable case). The sequence state variables, $s_{0:T}$, are responsible for generating the output of the system. The actual information stored in the state variable depends on the system being described. In addition to the state variable, all we need to define a system is a *state transition* function, *f*,

$$\begin{aligned} s_t &= f(s_{t-1}), \\ | s_t | &= N \end{aligned} \tag{1}$$

which describes how the state of the system changes over time. As time progresses through each time-step, the state variable will change to reflect the current time-slice. In practice, the definition of *f* can depend on a vector of some other constant parameters, *θ*.

$$s_t = f(s_{t-1}, \theta), \tag{2}$$

Notice that this is a recursive definition. We can then run a simulation of our system over T time-slices, (i.e. calculate the progression of states, $s_{0:T}$), by specifying the base case, s_0, and parameter vector *θ*.

System Observations

In cases where the *observed* output of a physical system is digitally sampled, the output does not correspond exactly to the set of state vectors. Instead, it is a function of the current state. In this case we add another 'layer' of variables, $y_{1:T}$, to represent the observations. The length of an observation vector, $| y_t |$, is given as M (M=1 in single variable case). Where the value of each observation is given by the *observation function, h*.

$$\begin{aligned} y_t &= h(s_t,\ \theta) \\ | y_t | &= M \end{aligned} \tag{3}$$

In this case we don't need an initial value, as the definition is not recursive. In summary:

$$\begin{aligned} s_0 &= \pi, \\ s_t &= f(s_{t-1}, \theta), \\ y_t &= h(s_t,\ \theta) \\ | s_t | &= N \\ | y_t | &= M \end{aligned}$$

The top layer of nodes, $s_{1:T}$, represent the state vector. The bottom layer, $y_{1:T}$, represents the observed data.

Table 2

Linear Dynamical Systems, (LDS's)

If we only consider models with linear transition functions we are dealing with *Linear Dynamical Systems* (LDS's). This linearity enables us to define the transition and observation functions using square transformation matrices A and C, respectively. The new equations are:

$$\begin{aligned} s_t &= A\, s_{t-1} \\ y_t &= C\, s_t, \quad \text{where} \end{aligned} \tag{4}$$

where A is a N×N matrix and C is a 1×M matrix. An example of a such a system is a damped sinusoid with a constant angular frequency ω. The definition of the state-space model follows in the damped oscillator example.

Adding some 'randomness'

The above definitions, given s_0 and θ, are enough to describe a fully deterministic model. This means that we can tell *with certainty* what values of $s_{0:T}$, $y_{0:T}$ will be: we can simply plug in the initial variables and calculate each time-slice. While a good start, this is not a realistic model of a sound wave. We need to model both noise in the state transitions, to account for minute frequency/damping variations, as well as observation noise. The latter models the noise introduced by digitizing the wave with recording hardware. This will create a more realistic, i.e. 'noisy', sound wave. We do this adding noise terms to both the state and observation vectors:

$$s_t = A\, s_{t-1} + w \tag{5}$$

$$y_t = C\, s_t + v \tag{6}$$

Modelling process noise

The process noise is defined as *white gaussian*. A noise is white when it is not correlated through time; its value at any time t is not dependant to any other time period. It's labeled as *gaussian* because it is drawn from a normal (gaussian) distribution. It particular, a zero-mean gaussian with diagonal covariance matrix Q:

$$w \sim \mathcal{N}\left([0 \quad 0], \begin{pmatrix} Q_{11} & 0 \\ 0 & Q_{22} \end{pmatrix}\right) \tag{7}$$

where $\mathcal{N}(\mu,\Sigma)$ represents a multivariate gaussian distribution with mean μ and covariance matrix Σ.
x ~ $\mathcal{N}(\mu,\Sigma)$ means that x is a random variable, the value of which is drawn from the distribution. We can see how the diagonal terms of Q add noise the state vector. We can simplify (10) to get:

$$\begin{aligned} s_t &= A\, s_{t-1} + \mathcal{N}(0, Q) \\ s_t &\sim \mathcal{N}(A\, s_{t-1}, Q) \end{aligned} \tag{8}$$

Modelling observation noise

The definition of the observation noise is identical to the process noise except for the dimensionality. We define a observation noise covariance matrix R and write:

$$\begin{aligned} y_t &= C\, s_t + \mathcal{N}(0, R) \\ y_t &\sim \mathcal{N}(C\, s_t, R) \end{aligned} \tag{9}$$

■ Modelling state transitions with conditional probability distributions

The noise terms now render the system model indeterministic. This means that we are no longer predict, with full certainty, the values of future state variables. Both the transition and observation functions are continuous variables represented as conditional probability distributions.

$$\theta = \{A, C, Q, R, \pi\}$$
$$\begin{aligned} s_0 &= \pi \\ s_t &\sim p(s_t \mid s_{t-1}) \equiv \mathcal{N}(A\, s_{t-1}, Q) \\ y_t &\sim p(y_t \mid s_t) \equiv \mathcal{N}(C\, s_t, R) \end{aligned}$$

Where θ represents the given parameter constants. π is also given as the initial state.

Table 3

The term $p(y_t \mid s_t)$ is known as the *likelihood* of the observation. It will be explained in later sections.

■ Hidden / Observed variables

It is important to label the variables in our model as *hidden* or *observed*. The observation terms, $y_{1:T}$, are obviously observed. The state vectors are hidden terms. They are considered *hidden* because we can't directly measure them in any way. We can only derive them past state variables. This is a big problem when we don't have the initial state vector, s_0, from which to calculate s_1..., then s_2; all we have are noisy state observations, $y_{1:T}$. How can we find the values of these hidden state variables? The answer is to find the joint distribution $p(s_{1:T}, y_{1:T})$ by utilizing the conditional independence structure of the graphical model. This is known as *filtering*. It is useful to first review Bayes Belief Networks.

Bayes Networks

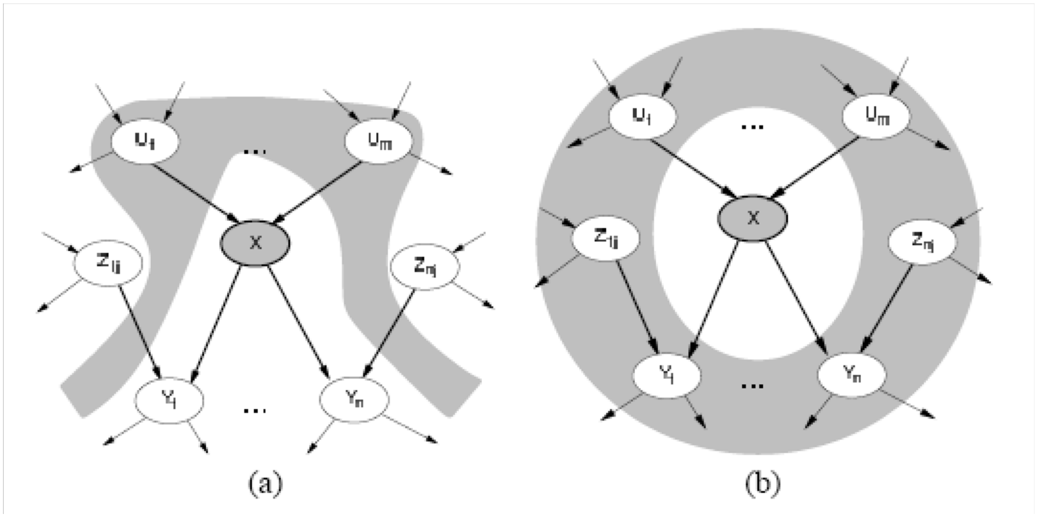
Definition

Probabilistic graphical models are directed graphs which represent joint probability distributions (Murphy, 2004). Arrows between nodes represent conditional probabilities; while lack of arrows represent conditional independance. More specifically, the value of each node, X, is a probabilty distribution conditioned on it's parent nodes, Pa(X).

$$p(X \mid \mathrm{Pa}(X)) = \prod_{i=1}^{N} p(x_i \mid \mathrm{Pa}(x_i)) \tag{10}$$

Conditional Independence, Local Markov Property

One real benefit of using a Bayes nets is the conditional independence assumption, i.e. that each node X is conditionally independant of its non-descendants given its parents, Pa(X). This is known as the *Local Markov Property.*



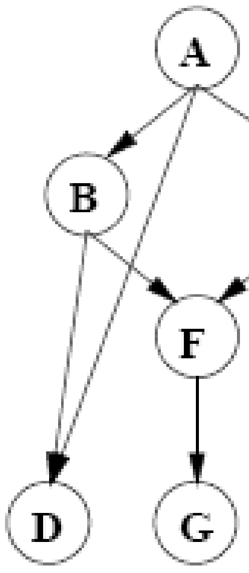
Local Markov Properties of Bayes nets. (a) A node X is conditionally independant of its nondescendants (e.g. Z_{1j}, ..., Z_{nj}) given its parents U_1, ..., U_m. (b) A node X is conditionally indepenendant of all other nodes in the network given its Markov blanket (shaded). From [RN02].

Figure 4

This representation allows calculation the joint probability distribution of all the nodes by using the *junction tree* algorithm (i.e. Filtering) (Murphy, 2004).

■ Filtering

The most common problem in Bayesian statistics is propagating a joint state distribution. This done by running 'forward' and 'backward' passes across the graph. For example, consider the graphical model in the next figure, (Murphy, 2004)



(Murphy, 2004)

Figure 5

The joint distribution is given by:

$$P(A,B,C,D,F,G) = P(A)P(B|A)P(C|A)P(D|B,A)P(F|B,C)P(G|F) \quad (11)$$

The conditional independance assumptions lead to the following factorization of the joint distribution.

$$\sum_{A,F,D,C,B,G} P(A,B,C,D,F,G) = \sum_{A} P(A) \sum_{F} \sum_{D} \sum_{C} P(C|A) \sum_{B} P(D|B,A) P(F|B,C) P(B|A) \sum_{G} P(G|F)$$

We can marginilize one variable at a time, working from right to left:

$$\sum_{A} P(A) \sum_{B} P(B|A) \sum_{C} P(C|A) \sum_{D} P(D|B,A) \sum_{F} P(F|B,C) \lambda_{G\to F}(F)$$

where $\lambda_{G\to F}(F) = \sum_G P(G\,|\,F)$.

The forward pass starts from the root node and propagates the forward 'message', π, by repeatadly calculating the conditional distributions for all child nodes. The backward 'message', λ, starts from the leaves and propagates the state distribution back to the root. Each step consists of marginalizing and conditioning. This is shown graphically in the next figure.

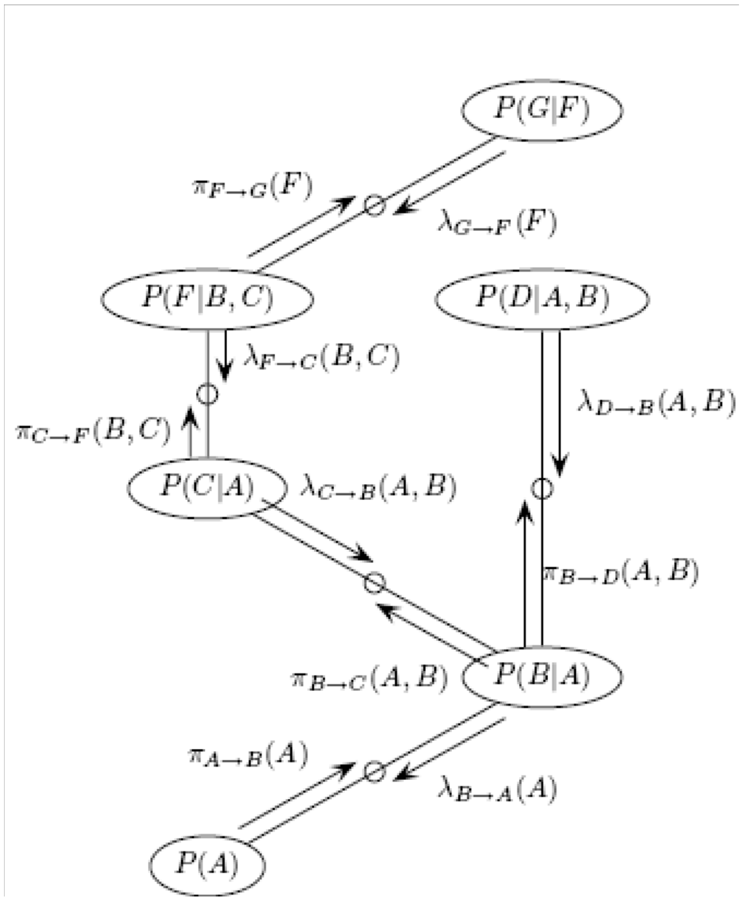


Figure 6

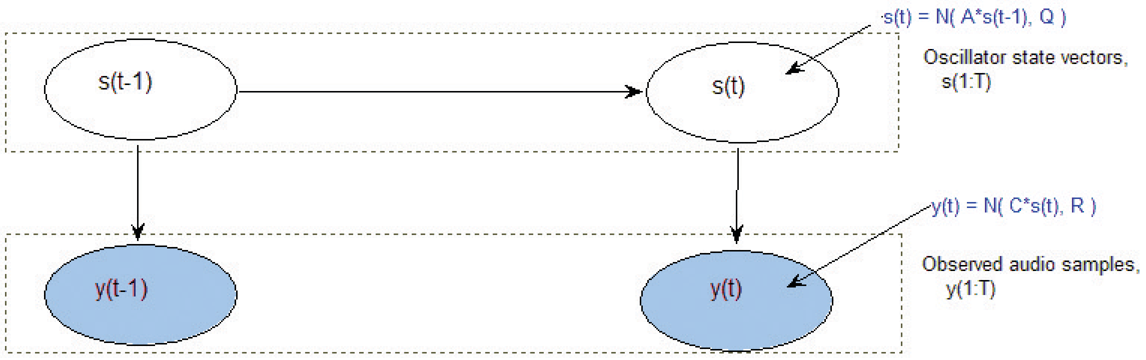
■ Dynamic Bayesian Networks, (DBN's)

A Bayes' net is a directed graph used to represent conditional probability distributions of nodes given their parent nodes. A dynamic Bayesian network (DBN) is a way to extend Bayes nets to model probability distributions over semi-infinite collections of random variables, Z_1, Z_2, ... (Murphy, 2004). We can partition the variables into Zt = (St, Yt) to represent the hidden and output variables of a state-space model (Murphy, 2004).

A DBN is defined to be a pair, $(B_1, B_\rightarrow)$, where B_1 is a BN which defines the prior $p(Z_1)$, and $B_\rightarrow$ is a two-slice temporal Bayes net (2TBN) which defines $p(Z_t \mid Z_{t-1})$ by means of a DAG (directed acyclic graph) as follows (Murphy, 2004):

$$p(Z_t \mid Z_{t-1}) = \prod_{i=1}^{N} p(Z_t^i \mid \mathrm{Pa}(Z_t^i)) \tag{12}$$

We must define the conditional probability distribution (CPD) of each node given its parents. For our model this includes $p(s_0)$, $p(s_t \mid s_{t-1})$ and $p(y_t \mid s_t)$; all are define above. The next figure depicts the DBN for our model:



The top layer of nodes, $s_{1:T}$, represent the state vector . The bottom layer, $y_{1:T}$, represents the observed data.

Figure 7

■ Estimating values of hidden variables, a.k.a. *filtering*

The answer is to *infer* the values of the hidden variables, $s_{0:T}$, conditioned on noisy observations. This is the most common problem in Bayesian statistics and is known as *filtering*. Think of filtering out the noise and returning the pure source of the observations.

Because the values of the hidden variables are unobserved and indeterministic, we can only make estimates, $\hat{s}_{1:T}$. The hat symbol, ^, indicates an estimated or predicted value.

Because we are making estimates of the hidden variables (a.k.a. *state estimate,* or *belief state*), we also associate a degree of belief with our estimate. This is the essence of Bayesian probability: we use probabilities to represent the certainty in an estimate. More specifically, the *belief state* at time t is a probability density (i.e. normalized distribution) over all possible states at time t given the observations, i.e. $p(s_{1:T} \mid y_{1:T})$. This is known as the *posterior distribution, or filtering density.* It can be calculated using an the prior state estimate, the *likelihood* of the evidence given the estimate, and Bayes' Theorem:

$$p(s_{1:T} \mid y_{1:T}) \equiv \frac{p(y_{1:T}|s_{1:T})\, p(s_{1:T})}{p(y_{1:T})} \tag{13}$$

Same equation with labeled terms is:

$$\text{Posterior} \equiv \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \tag{14}$$

The prior probability is propagated from the initial state estimate, $\pi \equiv p(s_0)$,

$$p(s_{1:T}) \equiv \pi\, p(s_0 \mid s_1)\, p(s_1 \mid s_2)\, p(s_2 \mid s_3)\, \ldots\, p(s_{T-1} \mid s_T) \tag{15}$$

The likelihood term is defined in the generative model.

The normalization factor, or *evidence*, is the sum of the marginal likelihoods of the data for all states s_t:

$$p(y_{1:T}) \equiv \int_{s_{1:T}} p(y_{1:T} \mid s_{1:T})\, p(s_{1:T}) \tag{16}$$

In general, $\sum$ is used to marginalize over discrete variables, $\int$ is used to marginalize continuous variables. For Linear Dynamical Systems, the marginilization can be done using the Kalman Filter equations.

Damped Oscillator Example

The generation of a digitally sampled damped sinusoidal of frequency ω can be stated as a linear dynamical system given by (Cemgil, 2004). We assume that our observations of the system are digital samples taken at a constant rate Fs.

Observations - $y_{1:T}$

At each time slice the observation represents a single audio sample. The value of y_t represents the amplitude of the wave at time t. It has only a single component, so M=1.

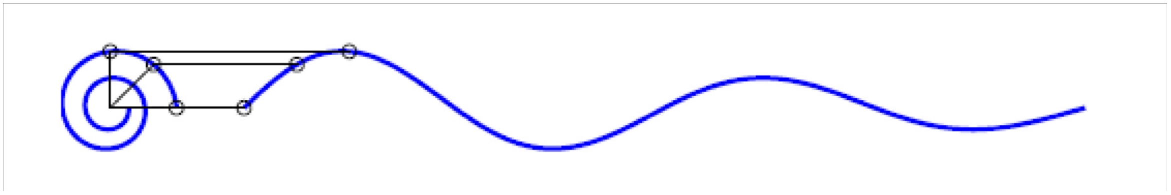
State Vector - $s_{1:T}$

The waves can be generated by a two-dimensional *oscillator* vector, s_t, which rotates around the origin with an angular frequency ω. The length of vector s_t corresponds to the initial amplitude of the sinusoidal and decreases over time at rate constant ρ. This is called the *damping coefficient*. Because the variables ω and ρ stay constant for a given model, they are parameters,

$$\theta = \{\omega, \rho\} \tag{17}$$

System parameters

The observations are generated by projecting the 2-dimensional state vectors onto a 1-dimensional plane (over time). This is depicted in the figure below (Cemgil, 2004).



A damped oscillator in state space form. Left: At each time step, the state vector s_t rotates by ω and its length becomes shorter. Right: The actual waveform is a one dimensional projection from the two dimensional state vector.

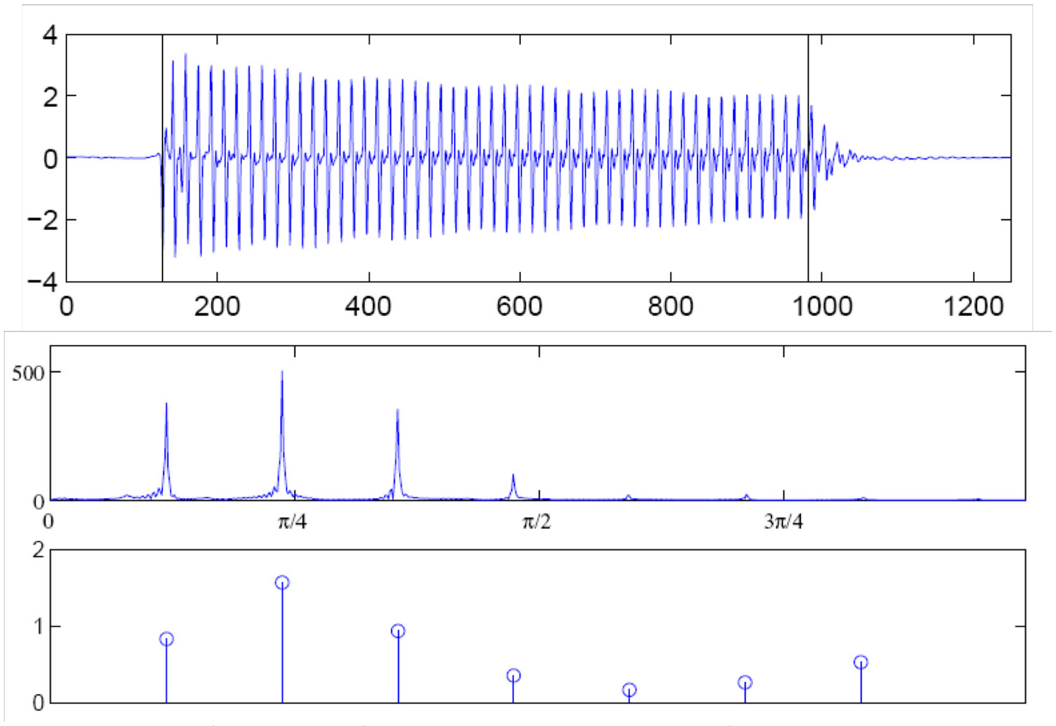
Figure 8

■ Initial state estimate - π

The initial state estimate, $\pi \equiv p(s_0)$ is drawn from a zero-mean gaussian with covariance matrix S.

$$\pi \sim \mathcal{N}(0, S) \tag{18}$$

The diagonal sum of S, Tr(S), is proportional to the initial amplitude of sound wave. The covariance structure defines how the 'energy' is distributed along the harmonics (Cemgil, 2004). It is initially estimated with a strong 1st and 2nd harmonics. The later overtones have increasingly less energy. This pattern can be shown by a Fourier transform of the audio signal.



Top: Audio signal of electric bass. Originally at 22 kHz, down-sampled by factor D=20. Middle: Frequency Spectrum (DFT) of audio signal Bottom: Diagonal entries of S for H=7 correspond to overtone amplitudes

Figure 9

Transition Matrix

The transition matrix, A, is responsible for rotating the state vector around the origin by ω and decreasing it's length by ρ (i.e. projecting $s_{t-1} \underset{A}{\Rightarrow} s_t$.). This is defined with Given's rotation matrix, B(ω), which rotates a two dimensional vector by ω radians:

$$B\ (\omega) \equiv \begin{pmatrix} \text{Cos}[\omega] & -\text{Sin}[\omega] \\ \text{Sin}[\omega] & \text{Cos}[\omega] \end{pmatrix} \tag{19}$$

Given' s rotation matrix.

$$A \equiv \rho\, B_\omega$$

$$A \equiv \begin{pmatrix} \rho\text{Cos}[\omega] & -\rho\text{Sin}[\omega] \\ \rho\text{Sin}[\omega] & \rho\text{Cos}[\omega] \end{pmatrix} \tag{20}$$

State transition matrix

Adding Harmonics

For a system modelling H harmonics, or overtones, we expand the state vector and transition matrices. The length of the state vector will now be 2H. It stores an (x,y) vector for each oscillator in the system; they are appended together.

The oscillator at harmonic h rotates at an angular frequency h ω and is damped with a coefficient ρ^h. The frequencies of overtones are discussed in an earlier section. The decreasing damping factor, (decreasing because $\rho < 1$, and $p_h = (p_{h-1})^2$, along the harmonics is consistent with real-life data from sampled instruments. The new transition matrix is defined as:

$$A \equiv \begin{pmatrix} \rho\text{Cos}[\omega] & -\rho\text{Sin}[\omega] & 0 & 0 & \cdots & \cdots & 0 & 0 \\ \rho\text{Sin}[\omega] & \rho\text{Cos}[\omega] & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & \rho^2\,\text{Cos}[2\,\omega] & -\rho^2\,\text{Sin}[2\,\omega] & \cdots & \cdots & 0 & 0 \\ 0 & 0 & \rho^2\,\text{Sin}[2\,\omega] & \rho^2\,\text{Cos}[2\,\omega] & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho^H\,\text{Cos}[H\,\omega] & -\rho^H\,\text{Sin}[H\,\omega] \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho^H\,\text{Sin}[H\,\omega] & \rho^H\,\text{Cos}[H\,\omega] \end{pmatrix} \tag{21}$$

■ Observation Matrix

The observations are simply projections of the state vector. We define the observation matrix as 1×2H dimensional projection matrix:

$$C = [0 \quad 1 \quad .... \quad 0 \quad 1] \qquad (22)$$

Observation matrix

■ Summary of generative model

$$\begin{aligned} \theta &= \{\omega, \rho, H, Q, R, \pi\} \\ s_0 &= \pi \\ s_t &\sim p(s_t \mid s_{t-1}) \equiv \mathcal{N}(A\, s_{t-1}, Q) \\ y_t &\sim p(y_t \mid s_t) \equiv \mathcal{N}(C\, s_t, R) \end{aligned}$$

Where θ represents the given parameter constants, as in (6). π is also given as the initial state.

Table 4

The term $p(y_t \mid s_t)$ is known as the *likelihood* of the observation. It will be explained in later sections.

Multinormal Distribution

Definition

Let $X \in \mathbb{R}^d$ represent a normally distributed random vector. Then we write,

$$\begin{aligned} X &\equiv [x_1, x_2, x_3, \ldots x_i, \ldots x_d] \\ X &\sim \mathcal{N}(\mu,\Sigma) \\ \mu &\equiv [\mu_1, \mu_2, \mu_3, \ldots \mu_i, \ldots \mu_d] \end{aligned} \tag{23}$$

μ represents the mean value and Σ represents the covariance matrix. Each index (i,j) in the covariance matrix holds the value cov(x_i, x_j).

Probability Density Function

The probability density function (PDF) of a multi-variate gaussian (multinormal) distribution is given by:

$$\begin{aligned} p(x) &= \mathcal{N}(\mu,\Sigma) \\ &= |\, 2\pi\Sigma\,|^{-\frac{1}{2}} \exp(-\tfrac{1}{2}\,(x-\mu)^{T}\,\Sigma^{-1}(x-\mu)) \end{aligned} \tag{24}$$

Mean - μ_x

The mean of multinormal distribution, X ~ $\mathcal{N}(\mu_x, \Sigma_x)$, is equivilent to the *expected* value of X.

$$\mu_x \equiv E(X) \equiv \int_i p(x_i)\, x_i \, dx \tag{25}$$

This is also known as the *first order moment* of p(x).

Variance

The variance is the *second order central moment* of the probability density function (Ahrendt, 2005).

$$\text{var}(X) \equiv E((X-\mu)^2) \tag{26}$$

- **Covariance - Σ_x**

The covariance is also a *second order moment,*

$$\operatorname{cov}(i, j) \equiv E((x_i - \mu_i)(x_j - \mu_j)) \tag{27}$$

The covariance matrix is an N×N matrix where $\Sigma_{i,j}$ holds the value $\operatorname{cov}(x_i, x_j)$.

- **Linear transformations of normally distributed random variables**

Let A, B $\in \mathbb{M}^{c*d}$ and c $\in \mathbb{R}^d$. Let X ~ $\mathcal{N}(\mu_x, \Sigma_x)$ and Y ~ $\mathcal{N}(\mu_y, \Sigma_y)$ be *conditionally independent* (Ahrendt, 2005).

$$\mathrm{Z} = \mathrm{AX} + \mathrm{BY} + \mathrm{c} \sim \mathrm{N}(\mathrm{AX} + \mathrm{BY} + \mathrm{c}, \mathrm{A}\Sigma_x\mathrm{A}^{\mathrm{T}} + \mathrm{B}\Sigma_x\mathrm{B}^{\mathrm{T}}) \tag{28}$$

- **Marginilization**

Let $[x_1, x_2, \ldots, x_c, x_{c+1}, \ldots, x_N] \sim \mathcal{N}(\mu_x, \Sigma_x)$. This is a *joint probability distribution, $p(x_{1:N})$*. If we with to find $p(x_{1:c})$, we have to *marginilize* over variables $x_{c+1:N}$,

$$p(x_1, \ldots, x_c) = \int \ldots \int_{\mathbb{R}^{c+1:N}} p(x_1, \ldots x_N)\, \mathrm{dx}_{c+1} \ldots x_N \tag{29}$$

Finding the mean and covarance of the marginal distribution is done by extracting the corresponding elements from the joint distribution (Arhendt, 2005).

$$p(x_1, \ldots, x_c) = p(x_{1:c}) = \mathcal{N}_{1:c}(\mu_{1:c}, \Sigma_{1:c}) \tag{30}$$

where $\mu_{1:c} = [\mu_1, \ldots, \mu_c]$ and

$$\Sigma_{1:c} = \begin{pmatrix} c_{11} & c_{21} & \cdots & c_{11} \\ c_{12} & c_{22} & \square & \vdots \\ \vdots & \square & \ddots & \vdots \\ c_{1\,c} & \cdots & \cdots & c_{\mathrm{cc}} \end{pmatrix} \tag{31}$$

- **Conditioning**

Let X ~ $\mathcal{N}(\mu_x, \Sigma_x)$ be represented with the following mean and covariance:

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \qquad \text{and} \qquad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \tag{32}$$

The conditional distribution, $p(x_1 \mid x_2)$, is also a normal distribution,

$$p(x_1 \mid x_2) \sim \mathcal{N}(\mu_1 + \Sigma_{12}\, \Sigma_{22}{}^{-1}(x_2 - \mu_x),\ \Sigma_{11} - \Sigma_{22}{}^{-1}\, \Sigma_{12}^{\mathrm{T}}) \tag{33}$$

Kalman Filter

System Model

The Kalman Filter is a set of equations which can be used to propagate the state estimate of a Linear Dynamical System with *white gaussian* noise. The observations of such a system result from the following equations:

$$s_t = A\, s_{t-1} + w \tag{34}$$

$$y_t = C\, s_t + v \tag{35}$$

Message Representation

Inference on Kalman Filter Models is equivilent to the junction tree algorithm; but is more complicated due to the continuous random variables. We propagate an estimate represented by the 1st and 2nd order statistical moments of the state distribution (Welch):

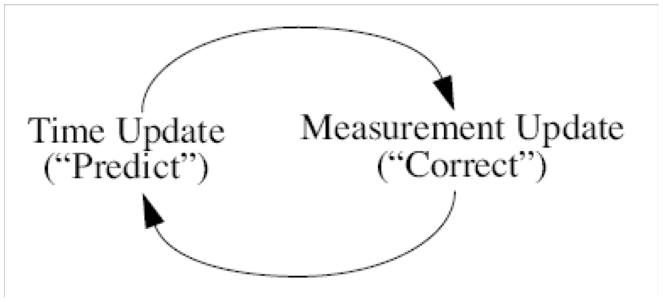
$$\begin{aligned} p(x_k \mid y_k) &\sim \mathcal{N}(E[x_k],\, E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^{\mathrm{T}}]) \\ &\equiv \mathcal{N}(\hat{x}_k,\, P_k) \end{aligned} \tag{36}$$

where,

$$\begin{aligned} \hat{x}_k &= E[x_k] \\ P_k &= E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^{\mathrm{T}}] \end{aligned} \tag{37}$$

Filtering Algorithm

From the recursive definitions above we see that posterior distribution is initially estimated and then propagated across time-slices. This is done in a cycle of prediction and correction equations.



The ongoing discrete Kalman filter cycle. The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time. Figure taken from (Welch, 2001)

Figure 10

Conditioned on the prior distribution from t-1 (or π in case of t=1), a prediction of the next audio sample (a.k.a. *a priori* estimate) can be calculated from the current belief state using the kalman prediction equations.

■ *a Priori* state estimate

The *a priori* estimate is a prediction. Consider it as a initial estimate 'before' observing y_t,

$$\delta_{t|t-1} \equiv \int_{s_{t-1}} p(s_t \mid s_{t-1},\ y_{1:t-1}), \tag{38}$$

The Kalman filter predict equations calculate a priori estimate as a *linear transformation* of the previous state distribution estimate. See (29):

$$\hat{s}_k^{\,-} = A\, s_{k-1} \tag{39}$$

$$P_k^{\,-} = A\, P_{k-1}\, A^{\mathrm{T}} + Q \tag{40}$$

This projection of the state vector across time slices, $s_{t-1} \overset{A}{\rightarrow} s_t$, corresponds to marginalization over s_{t-2}. This is due to the *Local Markov Property* (i.e. s_t is conditionally independent of $s_{t-2,}$ given s_{t-1}).

In the correct step, we use the *observation estimate,* $\hat{y}_{t|t-1}$, along with observed sample, y_t, to calculate the conditional likelihood of the observation.

■ *a Posteriori* state estimate

The *a posteriori* estimate is the result of the priori estimate conditioned on y_t.

$$\delta_{t|t-1} \equiv p(y_t \mid s_t)\, \delta_{t|t-1}, \tag{41}$$

This is calculated by adjusting the *priori* estimate using our predicted covariance and a weighted estimate error, defined next.

Estimate errors

Let $\hat{x}_k, \hat{x}_k^- \in \mathbb{R}^N$, represent the *a posteriori* and *a priori* state estimates and time-step k. The estimate errors are given as (Welch)

$$\begin{gathered} e_k^- \equiv x_k - \hat{x}_k^-, \text{ and} \\ e_k \equiv x_k - \hat{x}_k. \end{gathered} \tag{42}$$

The *a priori* estimate covariance is

$$P_k^- = E(e_k^- \, e_k^{-\mathrm{T}}) \tag{43}$$

The *a posteriori* estimate covariance is

$$P_k = E(e_k \, e_k^{\mathrm{T}}) \tag{44}$$

Kalman Filter Gain

Our goal is to propagate an estimate such that the *posterior* error covariance is minimized. At each time step, we update our previous estimate using the difference in our predicted measurement, $C\,\hat{x}_k^-$, and the actual observed measurement, y_k. This difference is known as the *residual*, or the *innovation* in the measurement.

The minimization is done by substituting equation (22) into the definition for e_k, (Welch)

$$\begin{aligned} e_k &\equiv x_k - (\hat{x}_k^- + K_k(y_k - C\,\hat{x}_k^-)) \\ &\equiv x_k - \hat{x}_k^- - K_k(y_k - C\,\hat{x}_k^-) \\ &\equiv e_k^- - K_k(y_k - C\,\hat{x}_k^-) \end{aligned} \tag{45}$$

substituting that into equation (18),

$$P_k = E((e_k^- - K_k(y_k - C\,\hat{x}_k^-)) \;\; (e_k^- - K_k(y_k - C\,\hat{x}_k^-))^{\mathrm{T}}) \tag{46}$$

performing the indicated expectations, taking the derivative of the trace of the result with respect to *K*, setting that result equal to zero, and then solving for *K,* known as the *blending factor: (Welch)*

$$K_k = P_k^- \, C^{\mathrm{T}} \, (C\,P_k^- \, C^{\mathrm{T}} + R)^{-1} \tag{47}$$

Correction Equations

The Kalman filter update equations:

$$\begin{gathered} K_k = P_k^- \, C^{\mathrm{T}} \, (C\,P_k^- \, C^{\mathrm{T}} + R)^{-1} \\ \hat{x}_k = \hat{x}_k^- + K_k(y_k - C\,\hat{x}_k^-) \\ P_k = (I - K_k\, C)\, P_k^- \end{gathered} \tag{48}$$

▪ What exactly is Likelihood?

The likelihood of a particular audio sample reflects the difference between the predicted audio sample, $\hat{y}_t$, and observed sample, y_t. This difference is known as the *residual*, or the *innovation*:

$$e = y_t - \hat{y}_t, \tag{49}$$

The likelihood is calculated as:

$$\mathcal{L} \equiv \mathcal{N}(e; 0, V_t) \tag{50}$$

As the residual increases, the likelihood of the configuration $r_{1:t}$ decreases at a rate inversely proportional to the state estimate covariance, P_t.

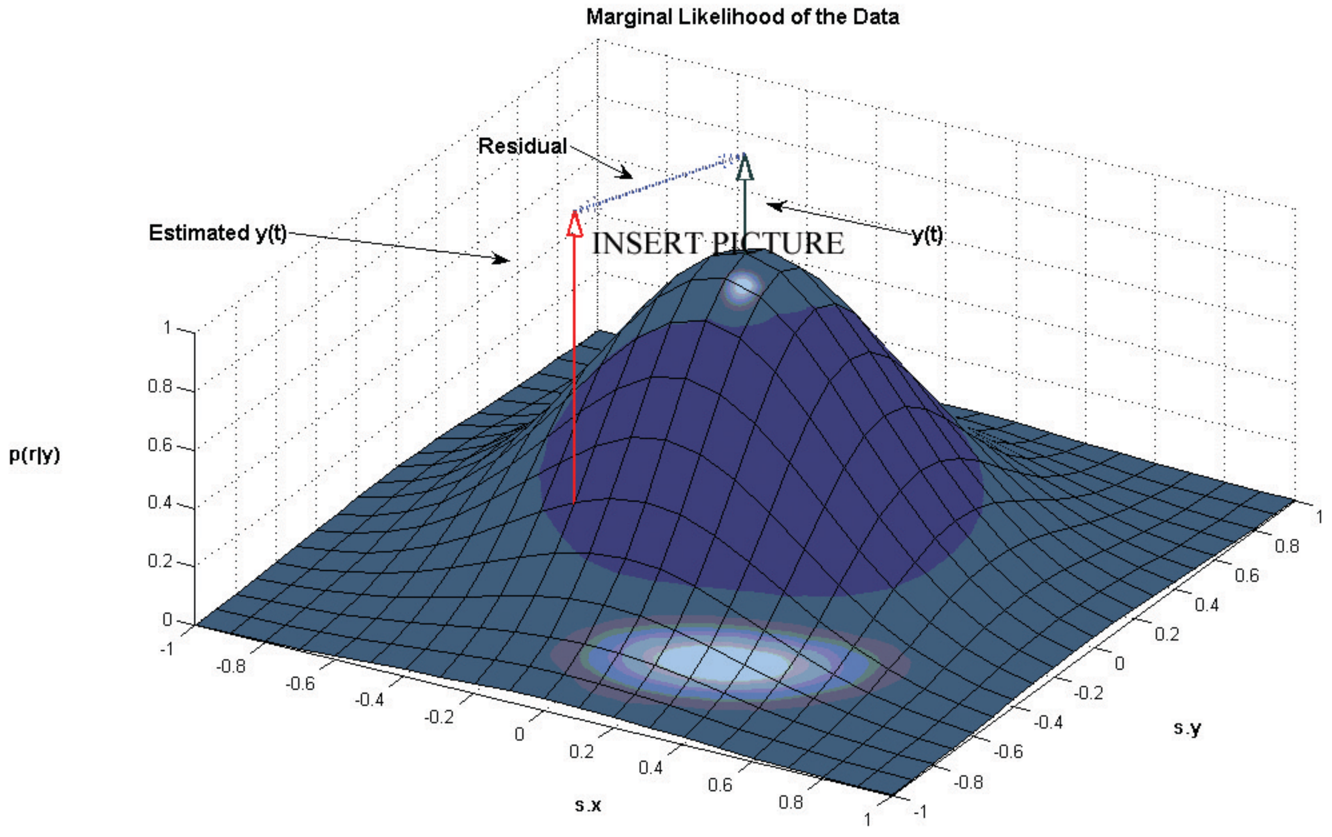


Figure 11

We can update our *a priori* estimate to an *a posteriori* estimate through conditioning on y_t. This corresponds to the *correction* Kalman Filter equations. The correction of the state vector depends on the state estimate covariance.

If the model is properly defined, our estimate covariance will decrease over time (i.e. we can be more confident in our estimate) as our state estimate converges on the source of the noisy samples. If it doesn't, we'll witness observation data with very low likelihoods. At this point we'll want to reconsider our parameter estimates.

Switching Kalman Filter Model for Monophonic Melodies

In this section the model is expanded to model a piano roll representation. The *piano roll*, or *score*, is a sequence indicators which label each time-slice with a state of *sound*, or *mute*. We define values *sound*=1 and *mute*=2 for simplicity.

Transition Matrices

We need to define transition matrices, $\{A_{\text{sound}}, A_{\text{mute}}\}$, for each state of the oscillator. They are the same as before except for the damping coefficient:

$$\begin{aligned} A_{\text{sound}} &\equiv \rho_{\text{sound}}\, B\,(\omega) \\ A_{\text{mute}} &\equiv \rho_{\text{mute}}\, B\,(\omega) \end{aligned} \tag{51}$$

ρ_{sound} is slightly less than 1. ρ_{mute} is much smaller than ρ_{sound}.

Defining the switch variables - $r_{1:T}$

The variables are referred to as *switch* variables because, at each time-step, they select which of two transition functions is used to project the next state. The two transition functions represent the *sound* and *mute* states:

$$r_{1:T} \quad \equiv \{\ r_t :\ r_t \in \{\text{sound} \wedge \text{mute}\},\quad 0 \le t \le T\} \tag{52}$$

For every 2-slice period, there are four possible combinations of $r_{t-1:t}$. The distribution of the configurations defines the prior on piano rolls. They can be organized as:

$$p(r_1) \quad \equiv \quad u(\{\text{sound, mute}\}) \quad \equiv \quad [.5,\ .5] \tag{53}$$

$$\begin{aligned} p\,(i,\,j) &\equiv p\,(r_t = i \mid r_{t-1} = j) \\ &\equiv \begin{Bmatrix} p(r_t = 1 \mid r_{t-1} = 1) & p(r_t = 1 \mid r_{t-1} = 2) \\ p(r_t = 2 \mid r_{t-1} = 1) & p(r_t = 2 \mid r_{t-1} = 2) \end{Bmatrix} \\ &\equiv \begin{pmatrix} 1 - 10^{-7} & 10^{-7} \\ 10^{-7} & 1 - 10^{-7} \end{pmatrix} \end{aligned} \tag{54}$$

The initial prior is a uniform distribution. The conditional distribution is a multinomial where it is much more likely for the switch variable to remain the same over time slices. We also define the function, isonset_t. It is true if there has been a note onset at the current time slice. A note onset is represented by the mute to sound transition. The updated model is as follows:

$$\theta = \{\omega, \rho_{\text{sound}}, \rho_{\text{mute}}, H\}$$

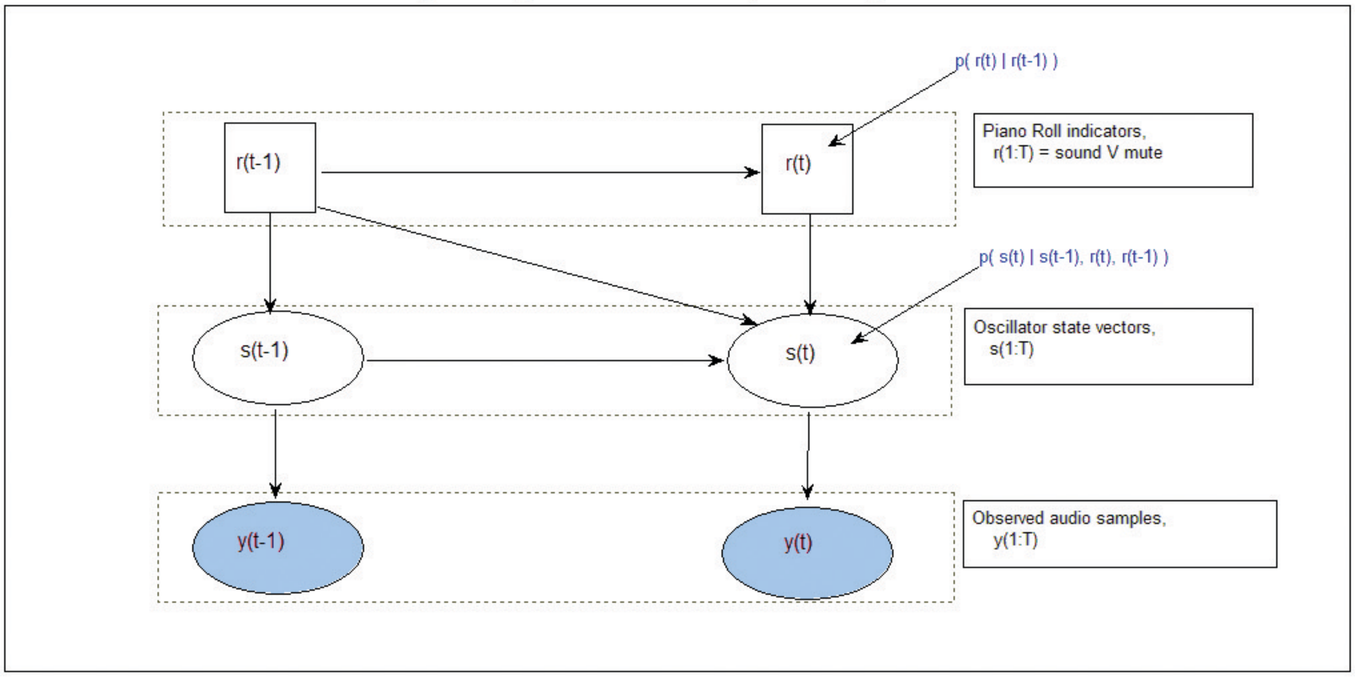
$$\text{isonset}_t \equiv [r_t = \text{sound} \wedge r_{t-1} = \text{mute}]$$

$$\begin{aligned} r_1 &\equiv u(\{\text{sound}, \text{mute}\}) \equiv \{.5, .5\} \\ r_t &\equiv p(i, j) \\ s_0 &\sim \mathcal{N}(0, S) \\ s_t &\sim \mathcal{N}(A_{r_t} s_{t-1}, Q) \\ y_t &\sim \mathcal{N}(C s_t, R) \end{aligned}$$

Table 5

Below is the DBN:

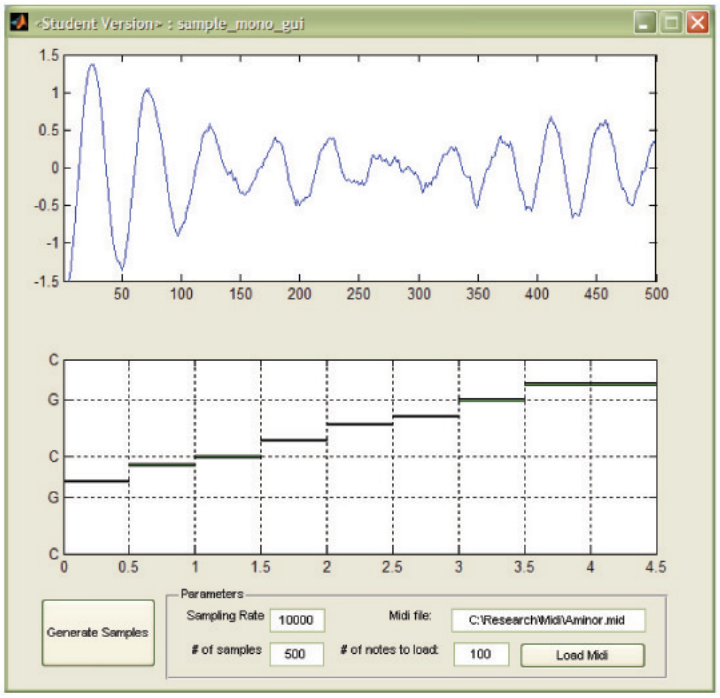
Dynamic Bayesian Network for Monophonic Melody Model


p(r(t) | r(t-1))
r(t-1)
r(t)
Piano Roll indicators, r(1:T) = sound V mute
p(s(t) | s(t-1), r(t), r(t-1))
s(t-1)
s(t)
Oscillator state vectors, s(1:T)
y(t-1)
y(t)
Observed audio samples, y(1:T)

The colored nodes are observed, white nodes are hidden. Discrete nodes are square, while continuous nodes are circular. Text in black font represents the names of each node. Text in blue font represents the value passed in δ(t)

▪ Model sample generation in Matlab

The following figure depicts samples taken from the model using Matlab.



Parameters
Sampling Rate 10000
Midi file: C:\Research\Midi\Aminor.mid
# of samples 500
# of notes to load: 100

Sample generator in Matlab

Figure 13

Inference for the Switching Kalman Filter model

Viterbi Estimation

Given an audio waveform, our goal is to infer the most likely sequence of piano roll indicators, $r^*_{1:T}$, which could give rise to the observed audio samples. This is, in general, a Viterbi estimation problem. The value we are seeking is defined as the *Maximum A Posteriori* trajectory:

$$r^*_{1:T} \equiv \underset{r_{1:T}}{\operatorname{argmax}}\, p(r_{1:T} \mid y_{1:T}) \tag{55}$$

It represents the most likely sequence of hidden variables to cause an observed output. This is different from the posterior distribution because it is just a point estimate. Calculation is the same as for filtering, except for replacing the summation over r_{t-2} with the maximization (MAP). It is important to note that, in Viterbi estimation, we propagate a *filtering potential,* $\delta_{1:T}$, as opposed to a *filtering density*, $\alpha_{1:T}$. The term potential used to indicate that this value is not normalized. Because we only care about the best piano roll, (i.e. configuration $r_{t:T}$ with highest likelihood), we can save calculations by using a point estimate.

$$p(r_{1:T} \mid y_{1:T}) \propto \underbrace{\int_{s_t} p(y_{1:t} \mid s_t, r_{1:t})}_{\text{filtering potential}} p(s_{1:t} \mid r_{1:t})\, p(r_{1:t}) \tag{56}$$

▪ Message Propagation

Inference is more difficult in the switching state-space model because we have to keep track of every possible enumeration of $r_{1:T}$ and return the sequence with the highest likelihood. The representing of the filtering potential is a Mixture of Gaussians (MoG) with a single component for each enumeration of $r_{1:T}$, (we use H=1 to simplify the examples):

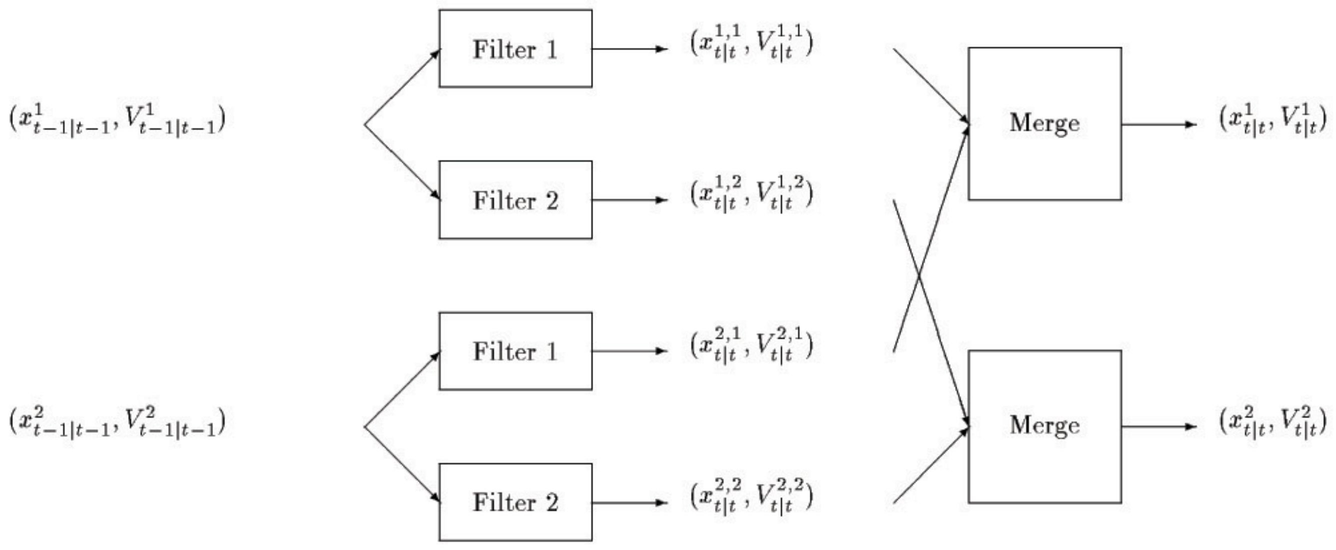
$$\delta_{t|t} \equiv p(y_{1:t}, s_t, r_t, r_{t-1}) \equiv \left\{ \begin{matrix} \delta_t(1, 1) & \delta_t(1, 2) \\ \delta_t(2, 1) & \delta_t(2, 2) \end{matrix} \right\} \quad (57)$$

where each,

$$\begin{aligned} \delta_{t|t}(i, j) &\equiv p(y_{1:t}, s_t, r_t = i, r_{t-1} = j) \\ &\equiv p(y_{1:t}, s_t, r_t = i, r_{t-1} = j) \end{aligned} \quad (58)$$

is also a MoG.

At each time step we make separate estimates for every configuration of piano roll indicators $\{r_t, r_{t-1}\}$. This corresponds to the prediction step. It can also be considered an *expand* step. As shown by the left side of the next figure.



$(x^1_{t-1|t-1}, V^1_{t-1|t-1})$
$(x^2_{t-1|t-1}, V^2_{t-1|t-1})$
Filter 1
Filter 2
Filter 1
Filter 2
$(x^{1,1}_{t|t}, V^{1,1}_{t|t})$
$(x^{1,2}_{t|t}, V^{1,2}_{t|t})$
$(x^{2,1}_{t|t}, V^{2,1}_{t|t})$
$(x^{2,2}_{t|t}, V^{2,2}_{t|t})$
Merge
Merge
$(x^1_{t|t}, V^1_{t|t})$
$(x^2_{t|t}, V^2_{t|t})$

Show Expand and Merge steps of message propagation on a Switching Kalman Filter

Figure 14

The correction/marginalization over r_{t-2} is done in the *merge* step. In the case of filtering, this is done with summation. In case of Viterbi, we use maximization.

■ Prediction

Upon receiving $\delta_{t-1|t-1}$ from the previous time slice,

$$\delta_{t-1|t-1} \equiv \left\{ \begin{matrix} \delta_{t-1|t-1}(1, 1) & \delta_{t-1|t-1}(1, 2) \\ \delta_{t-1|t-1}(2, 1) & \delta_{t-1|t-1}(2, 2) \end{matrix} \right\} \equiv p(y_{1:t-1}, s_{t-1}, r_{t-1}, r_{t-2}) \tag{59}$$

we with to calculate the *a priori* estimate, $\delta_{t|t-1}$, as follows.

First we marginalize over r_{t-2}. This is done by union over each row of $\delta_{t-1|t-1}$. These terms represent

$$\xi_{t-1}^{i} \equiv p(y_{1:t-1}, s_{t-1}, r_{t-1} = i) \equiv \sum_{r_{t-2}} p(y_{1:t-1}, s_{t-1}, r_{t-1} = i, r_{t-2}) \tag{60}$$

This marginalization corresponds to the 'merge' box in the above figure.

For components $\delta(1,1)$, $\delta(1,2)$, and $\delta(2,2)$ we next apply the transition model, $p(s_t \mid y_{1:t-1}, s_{t-1}, r_{t-1}, r_{t-2})$, using the Kalman Filter Time Update equations. We label each predicted potential as $\psi_{i,j}$.

$$\psi_{i,j} \equiv \int_{s_{t-1}} p(s_t \mid y_{1:t-1}, s_{t-1}, r_{t-1})\, \xi_{t-1}^{j}, \quad r_t = i \tag{61}$$

Multiplying each potential by the prior, $p(r_t \mid r_{t-1})$, gives:

$$p(y_{1:t-1}, s_t, r_t, r_{t-1}) \equiv p(y_{1:t-1}, s_t, r_{t-1}) p(r_t \mid r_{t-1}) \tag{62}$$

The component, $\delta_t(1, 2)$, is different because it represents a note onset. It's important to note that the piano roll configuration before the onset, $r_{1:\text{onset}}$, doesn't effect the likelihood of future indicators after it. This enables us to we can replace messages from ξ_{t-1}^{mute} with the maximum (scalar) likelihood estimate among them. We introduce this scalar value, Z_{t-1}^{mute}, as a prior for the next onset and tag the message with $r_{1:t-1}^{*}$. This replacement renders the algorithm tractable. The *a priori* potential is be given as:

$$\delta_{t|t-1} \equiv \left\{ \begin{matrix} p(1, 1)\, \psi_{1,1} & p(1, 2)\, Z_{t-1}^{\text{mute}}\, \xi_{t-1}^{j} \\ p(2, 1)\, \psi_{2,1} & p(2, 2)\, \psi_{2,2} \end{matrix} \right\} \tag{63}$$

■ Correction

Finally, the a Posteriori state estimate, $\delta_{t|t}$, is calculated using the Kalman Filter Correction equations on each component of the a *priori estimate:*

$$\delta_{t|t} \equiv p(y_{1:t}, s_t, r_t, r_{t-1}) \equiv p(y_t \mid s_t) p(y_{1:t-1}, s_t, r_t, r_{t-1}) \tag{64}$$

■ Forward Message Representation

To propagate the MAP trajectory, we define our forward-filtering message, δ_t, to store other variables in addition to the state estimate.

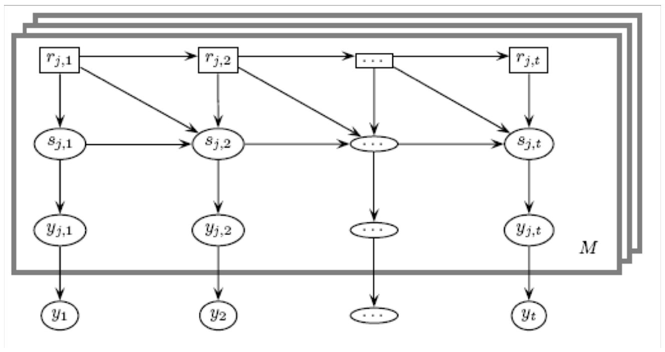
--Potential---

$\hat{s}_t$	$\equiv \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$, state estimate	
P_t	$\equiv$ 2x2 estimate cov.	
$P_{t\|t-1}$	$\equiv$ 2-slice state covariance matrix	
p_t	$\equiv p(r_{1:t})$	$\equiv$ prior probability
K_t	$\equiv$ Kalman Filter Gain	
V_t	$\equiv$ innovation covariance	
$y_t - \hat{y}_t$	$\equiv$ residual	
$\mathcal{L}$	$\equiv p(y_{1:t} \mid r_{1:t}) = \mathcal{N}(y_t; \hat{y}_t, V_t)$ $\equiv$likelihood	
$p(r_{1:t} \mid y_{1:t})$	$\propto \int_{s_t} p(y_{1:t} \mid s_t, r_{1:t})\, p(s_{1:t} \mid r_{1:t})\, p(r_{1:t})$	$\equiv$ posterior potential

Table 6

■ Extending the Model for multiple notes - $j_{1:T}$

We can simply expand the model to work for several notes at different frequencies. The frequencies are given, and are set to common MIDI values. We modify piano roll indicators $r_{1:T}$ to be $r_{1:T,1:M}$, where M is the number of notes in our model. Inference is the same except for indexing (Cemgil, 2004). The new DBN is as follows:



Graphical Model. The rectangle box denotes “plates”, M replications of the nodes inside. Each plate, j = 1, . . . ,M represents the sound generator (note) variables through time. (Cemgil, 2004)

Figure 15

■ Structure of prior distribution - $p(r_{t,j} \mid r_{t-1,j-1})$

The new prior distribution is given as:

$$p(r_{t,j} \mid r_{t-1,j-1}) = \begin{pmatrix} p_{1,1} & \square & \square & \square & p_{2,1}/M & \cdots & \cdots & p_{2,1}/M \\ \square & \ddots & \square & \square & \vdots & \ddots & \iddots & \vdots \\ \square & \square & \ddots & \square & \vdots & \iddots & \ddots & \vdots \\ \square & \square & \square & p_{1,1} & p_{2,1}/M & \cdots & \cdots & p_{2,1}/M \\ p_{2,1} & \square & \square & \square & p_{2,2} & \square & \square & \square \\ \square & \ddots & \square & \square & \square & \ddots & \square & \square \\ \square & \square & \ddots & \square & \square & \square & \ddots & \square \\ \square & \square & \square & p_{2,1} & \square & \square & \square & p_{2,2} \end{pmatrix}$$

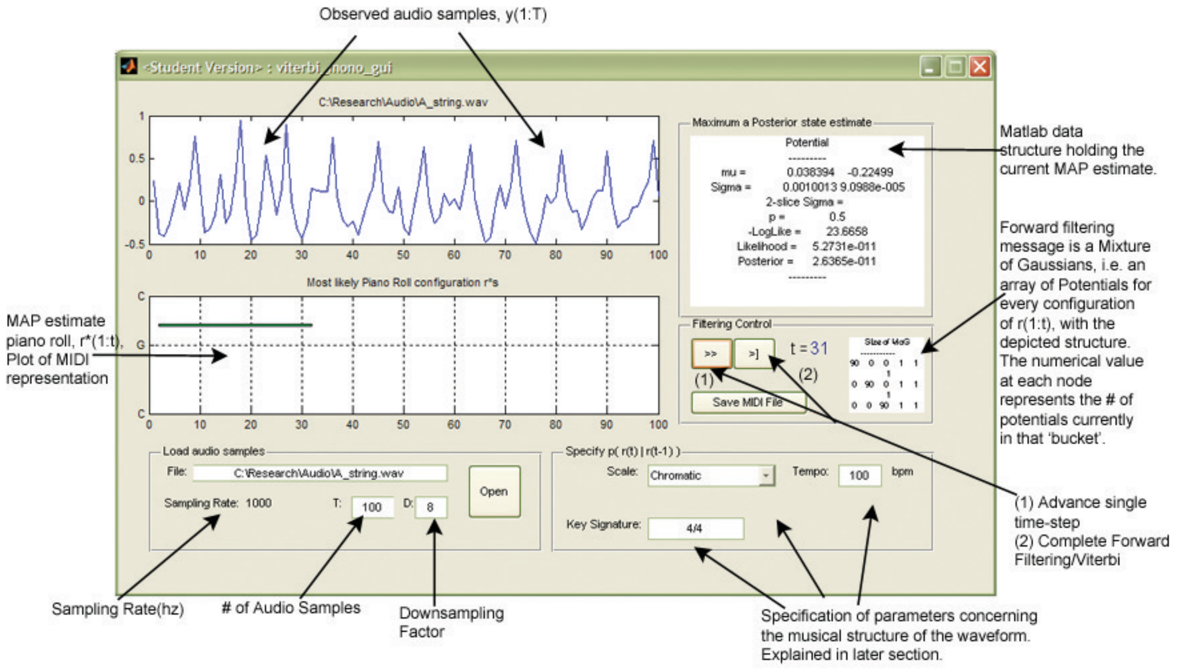
■ Structure of transition function - $f(r_{t,j} \mid r_{t-1,j-1})$

The new transition function is organized as follows:

$$f(r_{t,j} \mid r_{t-1,j-1}) = \begin{pmatrix} f_{1,1} & \square & \square & \square & f_{2,1}/M & \cdots & \cdots & f_{2,1}/M \\ \square & \ddots & \square & \square & \vdots & \ddots & \iddots & \vdots \\ \square & \square & \ddots & \square & \vdots & \iddots & \ddots & \vdots \\ \square & \square & \square & f_{1,1} & f_{2,1}/M & \cdots & \cdots & f_{2,1}/M \\ f_{2,1} & \square & \square & \square & f_{2,2} & \square & \square & \square \\ \square & \ddots & \square & \square & \square & \ddots & \square & \square \\ \square & \square & \ddots & \square & \square & \square & \ddots & \square \\ \square & \square & \square & f_{2,1} & \square & \square & \square & f_{2,2} \end{pmatrix}$$

■ Matlab Implementation

Below in a labeled screenshot of my Matlab implementation of Viterbi estimation.


Observed audio samples, y(1:T)
<Student Version> : viterbi_mono_gui
C:\Research\Audio\A_string.wav
Most likely Piano Roll configuration r*s
MAP estimate piano roll, r*(1:t), Plot of MIDI representation
Matlab data structure holding the current MAP estimate.
Forward filtering message is a Mixture of Gaussians, i.e. an array of Potentials for every configuration of r(1:t), with the depicted structure. The numerical value at each node represents the # of potentials currently in that 'bucket'.
(1) Advance single time-step
(2) Complete Forward Filtering/Viterbi
Sampling Rate(hz)
# of Audio Samples
Downsampling Factor
Specification of parameters concerning the musical structure of the waveform. Explained in later section.
Filtering Control
>>
>]
Save MIDI File
Open

Maximum a Posterior state estimate

```
         Potential
         ---------
mu =         0.038394   -0.22499
Sigma =      0.0010013 9.0988e-005
       2-slice Sigma =
      p =          0.5
   -LogLike =    23.6658
  Likelihood =   5.2731e-011
  Posterior =    2.6365e-011
         ---------
```

t = 31

```
   Size of MoG
   -----------
90   0   0   1   1
         1
0   90   0   1   1
         1
0    0  90   1   1
```

Load audio samples
File: C:\Research\Audio\A_string.wav
Sampling Rate: 1000 T: 100 D: 8

Specify p(r(t) | r(t-1))
Scale: Chromatic Tempo: 100 bpm
Key Signature: 4/4

Monophoni viterbi estimator in Matlab. The various controls and values are labeled.

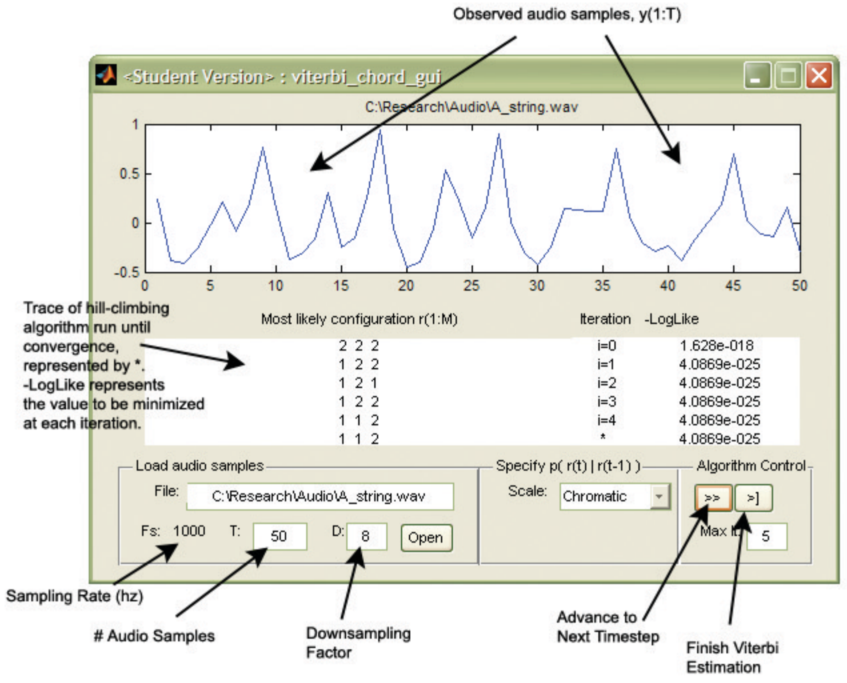
Figure 16

Polyphonic Chord Transcription

In this section we use the Viterbi estimation techniques learned in the last section use them to transcribe polyphonic chords. The difference in this case, is that we infer the variables $r_{1:M}$, rather than $r_{1:M,1:T}$. We assume that the configuration stays constant for t=0:T. This means that we have to work in windows if we wish to detect changes in chords over time.

The algorithm, given in (Cemgil, 2004), is a simple greedy search. We start with at an initial estimate of the chord configuration (zero's or drawn from prior), and prior $p(r_{1:M})$. A more informative prior can be very helpful in finding the correct configuration, $r_{1:M}$. We then calculate the likelihood of our estimate, α, and all configurations differing from our by a single note. If α is the most likely, we stop (at the local maximum). Otherwise we continue until convergence.

Using a uniform prior, $p(r_{1:M})$. We are (usually) able to find the correct chord configuration by taking the best of three trials using different initial estimates. This productivity can be improved by making more specifications in the prior, as explained in the next section. This is done by specifying some properties of the composition which generated the waveform. Next I show my Matlab implementation:


Observed audio samples, y(1:T)
<Student Version> : viterbi_chord_gui
C:\Research\Audio\A_string.wav
Trace of hill-climbing algorithm run until convergence, represented by *. -LogLike represents the value to be minimized at each iteration.
Load audio samples
Specify p(r(t) | r(t-1))
Algorithm Control
>>
>]
Open
Sampling Rate (hz)
# Audio Samples
Downsampling Factor
Advance to Next Timestep
Finish Viterbi Estimation

```
Most likely configuration r(1:M)        Iteration  -LogLike
          2 2 2                          i=0      1.628e-018
          1 2 2                          i=1      4.0869e-025
          1 2 1                          i=2      4.0869e-025
          1 2 2                          i=3      4.0869e-025
          1 1 2                          i=4      4.0869e-025
          1 1 2                           *       4.0869e-025
```

File: C:\Research\Audio\A_string.wav
Fs: 1000 T: 50 D: 8
Scale: Chromatic
Max It: 5

Matlab GUI for polyphonic chord transcription. The various controls and values are labeled.

Figure 17

Extending the Prior Distribution

The uniform prior over piano rolls/chord configurations leaves room for improvement. We can improve performance by specifying more information about the creation of the audio waveform. Because were are considering the use of this implementation mainly as a musical composition tool we can assume that the performer of the audio samples is available to supply the new parameters, which represent the details of the underlying composition.

■ Tonality in the prior

The parameters concerning the composition include the diatonic scale, Φ, the tempo, Δ, and the time-signature, Ξ. The diatonic scale labels each note, j_t, as a member or non-member. Non-members are relatively very unlikely, especially as the note duration increases. This is true because the use of these notes is usually limited to *passing* (transition) tones; they are rarely held ringing for a long duration. They are also more common on *offbeats* rather than *onbeats* (see Δ term). Members are additionally weighted by their index in the scale. For example, the I, IV, and V notes usually appear more commonly than others. In general, the probability of a note at time t is given by,

$$\mathrm{p}(\, j_t \mid j_{t-1}, \Phi, \Xi, \Delta \,) \tag{65}$$

The tempo term, Δ, labels each time-slice as *onbeats* or *offbeats,* given a specified quantization factor. In general, note onsets are much more common when they occur on or near a beat onset. The time-signature, Ξ, expands on this by considering exactly where a given note appears in the composition.

■ Physical instrument properties

We can also use physical properties of the guitar used to create the audio samples. This is possible because there are only a limited number of distinct fretboard *fingerings* for a given note/chord. For example, most notes appear at only 1-3 locations on the guitar fretboard, and each string can only generate a single simultaneous note.

For monophonic melodies, we define a fretboard-distance heuristic which weights the likelihood of a particular note by it's distance, in frets, from the previous note; the closer notes being more common. We also define notes appearing on adjacent strings as more likely.

For chords we use a similar heuristic by adding a weight to each configuration, proportional to the number of distinct hand *fingerings*.

Instrument Parameter Learning

To achieve reliable performance in real applications, our estimates of system model parameters, θ, can be learned for a particular instrument/recording setup. This is done by applying the standard EM (Expectation Maximization) algorithm to training data for each of M notes sampled from each string of the guitar.

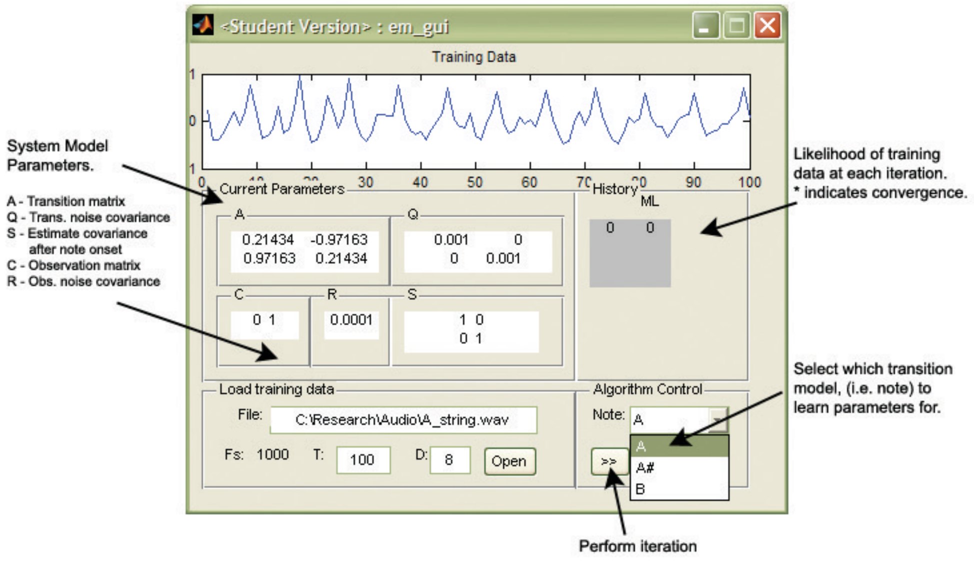
■ EM Algorithm

The EM algorithm iteratively maximizes the likelihood of a particular training set by calculating the derivative of the likelihood term, for each parameter, and adjusting appropriately. The likelihood of the training data is guaranteed to increase at every iteration until convergence. For a more complete description see (Murphy, 1998), (Digalakis, 1993), (Ghahramani, Z, 1996), and (Cemgil, 2004).

■ Training Data

We create the training data as follows. For each string, we sample the instrument playing each note at a regular interval and set tempo. For example, we could use a tempo of 100 bpm and play 12 ascending notes landing on beats. This would mean playing the first 12 frets on the guitar in ascending order. We then use an algorithm to split up each string waveform into sections, $y^{1:J}_{1:T}$, each containing a single note.

Each section is created such that the transition model remains constant. (i.e. note onset occurs at t=1 with j remaining constant, $r_{1:T,j}$=1). This permits us to treat each section as a *non-switching* linear dynamical system and calculate the likelihood using standard Kalman Filter equations. My Matlab implementation, with important values labels, is shown below.



<Student Version> : em_gui
Training Data
System Model Parameters.
A - Transition matrix
Q - Trans. noise covariance
S - Estimate covariance after note onset
C - Observation matrix
R - Obs. noise covariance
Likelihood of training data at each iteration.
* indicates convergence.
Select which transition model, (i.e. note) to learn parameters for.
Perform iteration
Current Parameters
History
ML
Load training data
Algorithm Control
Open
>>
A
A#
B

A: 0.21434 -0.97163; 0.97163 0.21434
Q: 0.001 0; 0 0.001
C: 0 1
R: 0.0001
S: 1 0; 0 1
ML: 0 0
File: C:\Research\Audio\A_string.wav
Fs: 1000 T: 100 D: 8
Note: A

The Matlab GUI for EM parameter learning.
The various controls and values are labeled.

Figure 18

- **EM Trace**

```
Training  Note : A5
iteration 1, loglik = -3663.604460
iteration 2, loglik = -311.786374
iteration 3, loglik = -9.922637
iteration 4, loglik = 96.145035
iteration 5, loglik = 145.776250
Training  Note : A#5
iteration 1, loglik = -2482.687963
iteration 2, loglik = 166.477802
iteration 3, loglik = 392.566403
iteration 4, loglik = 463.807662
iteration 5, loglik = 495.057545
Training  Note : B5
iteration 1, loglik = -5192.321380
iteration 2, loglik = 289.285348
iteration 3, loglik = 499.120020
iteration 4, loglik = 562.570657
iteration 5, loglik = 592.277587
Training  Note : C6
iteration 1, loglik = -5651.461128
iteration 2, loglik = 380.215932
iteration 3, loglik = 507.818473
iteration 4, loglik = 542.559441
iteration 5, loglik = 562.011764
Training  Note : C#6
iteration 1, loglik = -5846.257727
iteration 2, loglik = 555.151075
iteration 3, loglik = 692.551022
iteration 4, loglik = 713.507317
iteration 5, loglik = 725.033304
Training  Note : D6
iteration 1, loglik = -1920.004207
iteration 2, loglik = 367.811987
iteration 3, loglik = 416.248169
iteration 4, loglik = 445.989319
iteration 5, loglik = 467.426027
Training  Note : D#6
iteration 1, loglik = -2449.492102
iteration 2, loglik = 249.558678
iteration 3, loglik = 323.571290
iteration 4, loglik = 343.869571
iteration 5, loglik = 356.697474
```

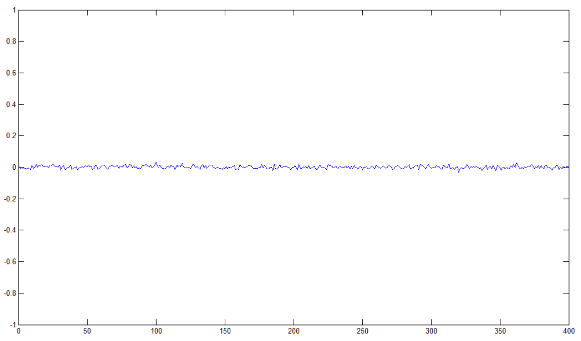
```
Training  Note : E6
iteration 1, loglik = -2843.839966
iteration 2, loglik = 647.360447
iteration 3, loglik = 695.314052
iteration 4, loglik = 704.969120
iteration 5, loglik = 711.679438
Training  Note : F6
iteration 1, loglik = -1176.609238
iteration 2, loglik = 584.270071
iteration 3, loglik = 673.051130
iteration 4, loglik = 694.098511
iteration 5, loglik = 709.990617
Training  Note : F#6
iteration 1, loglik = -662.881257
iteration 2, loglik = 546.030383
iteration 3, loglik = 628.870297
iteration 4, loglik = 651.425956
iteration 5, loglik = 668.676303
Training  Note : G6
iteration 1, loglik = -3337.526830
iteration 2, loglik = 211.806904
iteration 3, loglik = 282.428182
iteration 4, loglik = 310.866185
iteration 5, loglik = 329.663250
Training  Note : G#6
iteration 1, loglik = -408.300460
iteration 2, loglik = 420.828378
iteration 3, loglik = 472.637400
iteration 4, loglik = 489.918975
iteration 5, loglik = 504.464328
```

Polyphonic transcription experiment results.

The following sections show my results from chord transcription experiments. The best chord configuration for each iteration is represented as a boolean array of length M (the number of notes in the model). A listing of actual note names which make up the chord (chord tones) are included after the boolean array. Asterisks indicate a correct estimate.

■ Synthetic Samples

■ 3 notes, 1 Harmonic, 4096 hz, 400 samples

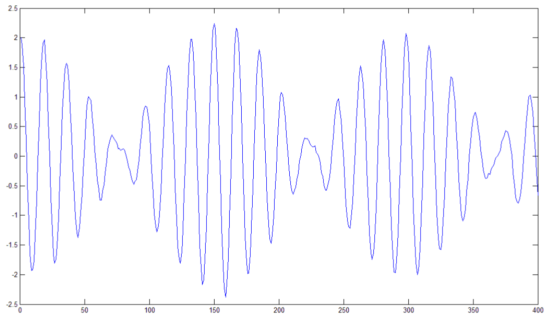


■ Chord 1

```
             Actual chord configuration                chord tones
             --------------------------------                ---------------
                        2 2 2                               []

                     Chord Configuration               Log Likelihood
                 ---------------------------------------      --------------------
i = 0                     2    2    2                    1143.2952      **
```

- **Chord 2**

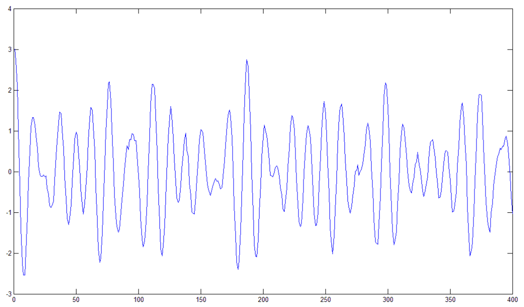


Actual chord configuration	chord tones
1 2 1	{A, B}

	Chord Configuration			Log Likelihood
i = 0	2	2	2	-474058.7542
i = 2	2	1	2	-1323.5236
i = 3	1	1	2	94.1418
i = 4	1	1	1	715.8884
i = 5	1	2	1	750.4153 **

- **12 notes, 1 Harmonic, 4096 hz, 400 samples**

- **Chord 1**



Actual chord configuration	chord tones
1 2 2 1 2 2 2 1 2 2 2 2	[A, C, E]

	Chord Configuration	-Log Likelihood
i = 0	2 2 2 2 2 2 2 2 2 2 2 2	-156933.7858
i = 1	2 2 2 2 1 2 2 2 2 2 2 2	-18761.5701
i = 2	2 2 2 2 1 2 2 1 2 2 2 2	-6361.4038
i = 3	1 2 2 2 1 2 2 1 2 2 2 2	-332.5287
i = 4	1 2 2 1 1 2 2 1 2 2 2 2	273.0216
i = 5	1 2 2 1 2 2 2 1 2 2 2 2	287.7244 **

- **Chord 2**

Actual chord configuration	chord tones
1 2 1 2 2 2 2 1 2 2 2 2	[A, Cb, E]

	Chord Configuration	-Log Likelihood
i = 0	2 2 2 2 2 2 2 2 2 2 2 2	-130687.7854
i = 1	2 2 2 1 2 2 2 2 2 2 2 2	-13959.8498
i = 2	2 2 2 1 2 2 2 1 2 2 2 2	-3437.5141
i = 3	1 2 2 1 2 2 2 1 2 2 2 2	-306.4038
i = 4	1 2 1 1 2 2 2 1 2 2 2 2	267.1139
i = 5	1 2 1 2 2 2 2 1 2 2 2 2	288.6613 **

■ Real Samples, Estimated Parameters

Estimated parameters give very poor results; none of the chords is identified correctly. 8128--->4096 indicates that the wave file has been downsampled by a factor of 2.

■ 12 notes, 8 Harmonic, 8128 ---> 4096 hz, 800 samples

■ Chord 1

```
              Actual chord configuration                    chord tones                              -
               -------------------------------               ---------------
1   2   2   2   2   2   2   2   2   2   2   2                  [A]

                   Chord Configuration              -Log Likelihood
               ---------------------------------------      --------------------
i = 0            2  2  2  2  2  2  2  2  2  2  2  2          -7208.0208
i = 1            1  2  2  2  2  2  2  2  2  2  2  2         -3151.3079
i = 2            1  1  2  2  2  2  2  2  2  2  2  2         -2793.4783
i = 3            1  1  1  2  2  2  2  2  2  2  2  2         -2700.2218
```

- **Chord 2**

Actual chord configuration	chord tones
2 2 2 1 2 2 2 2 2 2 2 2	[C]

	Chord Configuration	-Log Likelihood
i = 0	2 2 2 1 2 2 2 2 2 2 2 2	-4167.4467
i = 1	1 2 2 2 2 2 2 2 2 2 2 2	-922.0286
i = 2	1 1 2 2 2 2 2 2 2 2 2 2	-485.0826
i = 3	1 1 1 2 2 2 2 2 2 2 2 2	-312.9073
i = 4	1 1 1 1 2 2 2 2 2 2 2 2	-234.0392
i = 5	1 1 1 1 1 2 2 2 2 2 2 2	-194.6421
i = 6	1 1 1 1 1 1 2 2 2 2 2 2	-177.0988
i = 7	1 1 1 1 1 1 1 2 2 2 2 2	-174.1099

- **Chord 3**

Actual chord configuration	chord tones
1 2 1 2 2 2 2 1 2 2 2 2	[A, B, E]

	Chord Configuration	-Log Likelihood
i = 0	2 2 2 2 2 2 2 2 2 2 2 2	-18583.957
i = 1	1 2 2 2 2 2 2 2 2 2 2 2	-6373.7208
i = 2	1 1 2 2 2 2 2 2 2 2 2 2	-5087.4816
i = 3	1 1 1 2 2 2 2 2 2 2 2 2	-4609.0304
i = 4	1 1 1 1 2 2 2 2 2 2 2 2	-4418.8948
i = 5	1 1 1 1 1 2 2 2 2 2 2 2	-4365.9024

■ Real Samples, Learned Parameters

Results improve greatly after training. Results from EM iterations are shown after chords.

■ 12 notes, 8 Harmonic, 8128 ---> 4096 hz, 800 samples

■ Chord 1

```
              Actual chord configuration                    chord tones
              --------------------------------              ---------------
2   2   1   2   2   2   2   2   2   2   2   2               [B]

                    Chord Configuration              -Log Likelihood
                  ---------------------------------------     --------------------
i = 0               2  2  2  2  2  2  2  2  2  2  2  2        -4167.4467
i = 1               2  2  1  2  2  2  2  2  2  2  2  2        502.5977 **
```

■ Chord 2

```
              Actual chord configuration                    chord tones
                --------------------------------                      ---------------
2   2   2   1   2   2   2   2   2   2   2   2               [C]

                    Chord Configuration              -Log Likelihood
                  ---------------------------------------     --------------------
i = 0               2  2  2  2  2  2  2  2  2  2  2  2       -18583.957
i = 1               2  2  1  2  2  2  2  2  2  2  2  2        431.4149 **
```

■ Chord 3

```
              Actual chord configuration                    chord tones
               --------------------------------             ---------------
1   2   1   2   2   2   2   2   2   2   2   2               [A,B]

                    Chord Configuration              -Log Likelihood
                  ---------------------------------------     --------------------
i = 0               2  2  2  2  2  2  2  2  2  2  2  2            -4167.4467
i = 1               1  2  2  2  2  2  2  2  2  2  2  2            -72.6889
i = 2               1  2  1  2  2  2  2  2  2  2  2  2            13.696 **
```

Future Work

Future work includes implementation of the mentioned tonal anointments to the prior probability. Also, the prior could be learned by studying databases of music (Cemgil). Next, the implementation of the algorithms mentioned as a VST (Virtual Studio Technology) plug-in will be a useful tool for composing music.

Bibliography

[1] Ahrendt, Peter. *The Multivariate Gaussian Probability Distribution*. IMM, Technical University of Denmark

[2] Barber, D. (2004). *A Stable Switching Kalman Smoother*. In IDIAP-RR 04-89.

[3] Cemgil, A. T., Kappen, H. J., & Barber, D. (2003). *Generative model based polyphonic music transcription*. In *Proc. of IEEE WASPAA*, New Paltz, NY. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.

[4] Cemgil, A. T. *Bayesian Music Transcription*. PhD thesis, Radboud University of Nijmegen, 2004.

[5] V. Digalakis, J. R. Rohlicek, and M. Ostendorf. *ML estimation of a stochastic linear systemswith the EM algorithm and its application to speech recognition*. *IEEE Trans. on Speech and Audio Proc.*, 1(4):421-442, 1993.

[6] Doucet, A., de Freitas, N., K. Murphy, and S. Russell. *Rao-blackwellised particle filtering for dynamic Bayesian networks*. In *UAI*, 2000.

[7] Fearnhead, P. (2003). *Exact and efficient bayesian inference for multiple changepoint problems.* Tech. rep., Dept. of Math. and Stat., Lancaster University.

[8] Ghahramani, Z., & Hinton, G. E. (1996). *Parameter estimation for linear dynamical systems*. (crg-tr-96-2). Tech. rep., University of Totronto. Dept. of Computer Science.

[9] Kalman, R. E. (1960). *A new approach to linear filtering and prediction problems*. Transaction of the ASME-Journal of Basic Engineering, 35-45.

[10] Kedzierski, Mark (2004). *Real-Time Monophonic Melody Transcription*. Undergraduate research rep, University of Texas at Austin, Deparment of Computer Science.

[11] MacKay, D. J. C. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

[12] Moore, Brian C.J. (1997). *An Introduction to the Psychology of Hearing 4th Edition*, (Academic Press, London).

[13] Murphy, K. P. (1998). *Switching Kalman filters*. Tech. rep., Dept. of Computer Science, University of California, Berkeley.

[14] Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley.

[15] Raphael, C., & Stoddard, J. (2003). *Harmonic analysis with probabilistic graphical models*. In *Proc. ISMIR*.

[16] Welch, G., & Bishop, G. (2001). *An Introduction to the Kalman Filter*. SIGGRAPH course 8, University of North Carolina at Chapel Hill.