Finding Rhythm in Music Using Kalman Filters Optimized with Local Search

George Woskob - AA228 Fall 2019 - Dec 6, 2019

Examine the code at https://github.com/aphera/aa228-final-project

***Abstract*— Beats are the fundamental unit of time in music. They can be felt easily by many people as when they tap their feet to music. Beats are not directly observed but the notes that are played according to the periodicity of the beat are. Using the observable musical events of these notes, we can model our belief about the beat as a gaussian distribution. The observable beats can also be modeled according to gaussian distributions. As such, our belief about the beat can be updated over time using a Kalman filter. In this paper we update the parameters of the Kalman filter offline using local search (gradient ascent) to improve performance of the filter. The algorithm was designed to operate in real time as events are observed, updating the belief.**

I. NOMENCLATURE

Beat = A basic unit of time in music.

Musical event = A moment in time that is observable through the sound of a note played on an instrument.

Tempo = The speed at which music is played. A common measure is beats per minute (bpm). A faster tempo corresponds to a higher bpm as

BPM = Beats per minute. A common measure of tempo. For reference, modern dance music often falls at around 120 bpm.

II. INTRODUCTION

Music is periodic (rhythmic) and its speed is measured by beats, a unit of time in music around which musical events are organized. What we refer to as the beat is generally felt by people as they tap their foot to music or clap along to music. We infer the beat by observing musical events which generally correspond to notes played by instruments, whether that be harmonic in nature (a note bowed by a violin) or purely rhythmic (the striking of a drum). In music the events do not necessarily fall directly on the beat, though they will often do so. Events will also occur at frequencies that are at integer subdivisions or multiples of integer subdivision of the beat, among other places.

The problem this paper explores is beat detection given a series of musical events, that is, determining the length of a beat given the observable notes and the spaces between these notes. Beats are not directly observable and thus we can model the problem as a POMDP.

The duration of a beat can be represented as a gaussian distribution over the duration of a piece of music. Our belief about that beat given the observed musical events can also be represented as a gaussian distribution over time. As such, a Kalman filter can be used to update our belief given some observations. In this paper, however, we did not rely on statistical analysis to determine the parameters of the Kalman filter, rather we used gradient ascent as reinforcement learning to tune the parameters of the Kalman filter to improve performance of the beat detection algorithm. Bach's music was the source of data for the learning of the Kalman filter parameters. Much of his music has been impeccably digitized in the form of MIDI files, a format that encodes the notes, the durations between them, and the underlying tempo. As the tempo of the music is encoded in the file itself, we thus have our labeled training data.

What this paper does not attempt to do

In this paper we do not attempt to identify musical events from an audio source. That is, the algorithms defined here do not process any representation of sound. MIDI does not encode actual sound. It is more akin to a score, or sheet music. The algorithms assume that the location of the musical event in time has already been determined (as represented in the MIDI file).

III. RELATED WORK

A cursory search of related work in the field does not return any results showing that solving this problem has been attempted using a Kalman filter. Much of the work in the field uses discrete wavelet transforms to identify musical events then determine the periodicity using resonance filters[1][2].

IV. APPROACH

A. The Kalman Filter

The problem of determining the length of the beat was modeled as a POMDP. The underlying beat of a piece of music is not directly observable but the musical events which occur at frequencies that are at integer subdivisions or multiples of integer subdivision of the beat are observable.

The state of the problem is the bpm of the music. After each observable event in the music we can update the bpm

accordingly. For this process there are no actions or rewards. The observation is the time since the last musical event occurred.

The Kalman filter was implemented as such. On each observation the state is updated according to the transition function. In this case because x represents bpm, the transition function is the identity function because the bpm is assumed to remain constant.

$$x = x \cdot h \tag{1}$$

The variance of the state belief, p, is updated by the parameter q multiplied by our observation t, the time since the last observation. This represents the growth in uncertainty of our belief over time.

$$p = p + q \cdot t \tag{2}$$

Then, we naively infer a bpm given t

$$naïve\_bpm = 60 / t \tag{3}$$

and multiply that naïve bpm by an observation vector o_v defined by interval of some even numbered subdivision of 1 over some span i.e. `[0.125, .25, .5 … 1.75, 2]`. The intuition behind this is that any of our observations t could represent a duration that is proportional to any of these subdivisions of the actual duration of the beat. We then get our scaled observation vector, s_o_v.

$$s\_o\_v = o\_v \cdot naïve\_bpm \tag{4}$$

We then find the closest member of s_o_v to our current belief, x and this becomes our observation of the current bpm, z.

$$i = argmin(|sov[i] - x|) \tag{5}$$

$$z = sov[i] \tag{6}$$

We then need to calculate a number to represent the variance, r, of this value, z. We use that same i to pull a weight out of another vector, our observation weight vector o_w_v and add this weight to the squared error of z and x multiplied by some other error weight, e_w.

$$r = o\_w\_v[i] + e\_w \cdot (x - z)^2 \tag{7}$$

We then determine a simplified Kalman gain (K) (the actual Kalman filter implementation Kalman gain has more terms, but we can factor some out).

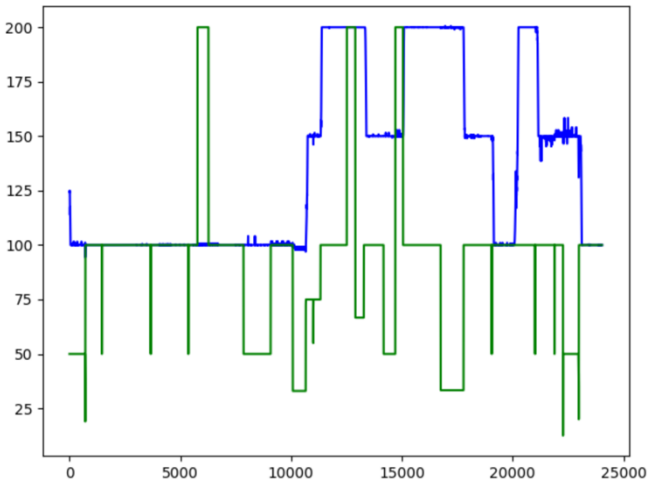
$$K = p \cdot h^T \cdot (h \cdot p \cdot h^T + r)^{-1} \tag{8}$$

Finally we update our state and variance.

$$x = x + K \cdot (z - h \cdot x) \tag{9}$$

$$p = p - K \cdot h \cdot p \tag{10}$$

At the beginning of the algorithm the state values x and p were set to 120.0 and 160.0 accordingly. The parameters of q, the components of the observation weight vector (o_w_v), and the error weight (e_w) were all set to 1. The observation vector (o_v) was set as previously described. Off the shelf this worked very given these default values.



Default parameters: The green line represents the true bpm of the entire *Goldberg Variations* as encoded in the MIDI files and in blue is the estimates of the bpm based on the results of the default parameters in our Kalman filter. On the y axis is the bpm, on the x axis is time in seconds.

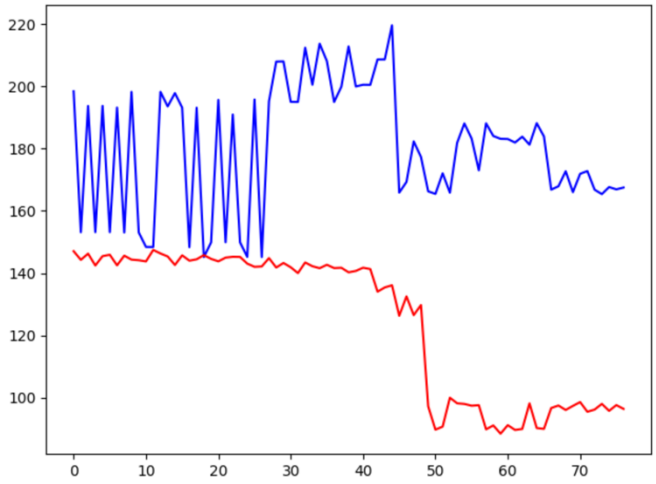
The mean square error of our estimate is 142.8

While the state values get updated as the algorithm runs, the parameters do not. In order to improve performance of the algorithm, we used gradient ascent to find better values for the parameters to the Kalman filter. Given each MIDI file encodes the true bpm of the given portion of the music, we could calculate the error of the estimate and try to minimize this. This became an MDP problem.

B. Gradient Ascent

The MDP for optimizing the Kalman filter parameters, q, the individual componenets of the observation weight vector (o_w_v), and the error weight (e_w) is defined as such: the state is the value for these parameters, the action is some amount by which we can increment or decrement one of the parameters or its components, the transition function is a deterministic application of this action, and the reward function is determined by the minimization of the error. The gradient ascent was run for 77 iterations with a variable number of steps used for the action each time, ranging between 1.0 and 0.1. In each iteration we updated a single parameter to the Kalman filter and found the set of parameters that reduced the error the most. In the following step we used the updated parameters as the starting point. I ran the gradient ascent against the second book of Bach's *The Well Tempered Clavier*, randomly sorting the pieces in that collection

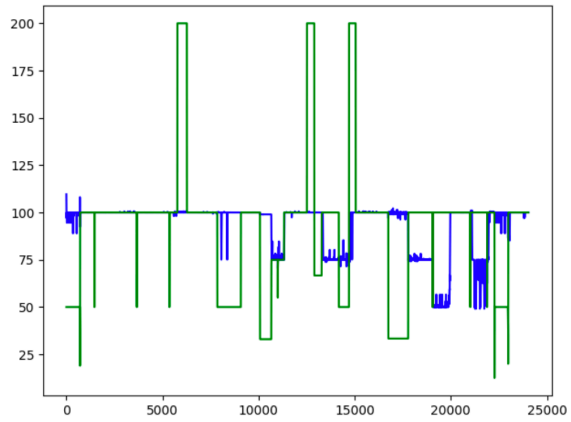
to hopefully introduce a slight amount of variation between iterations to prevent overfitting. Each time I also computed the error against the entire *Goldberg Variations*.



Performance during gradient ascent: The blue line represents the error of the estimate of the training data, book two of *The Well Tempered Clavier*. In each iteration the order of the pieces in the collection was shuffled which accounts for the fact that the gradient does not descend every time. In general it does. The red line represents the error of the estimate for the *Goldberg Variations* which despite not being used for the gradient ascent improved in a way that is much more pronounced that the training data.

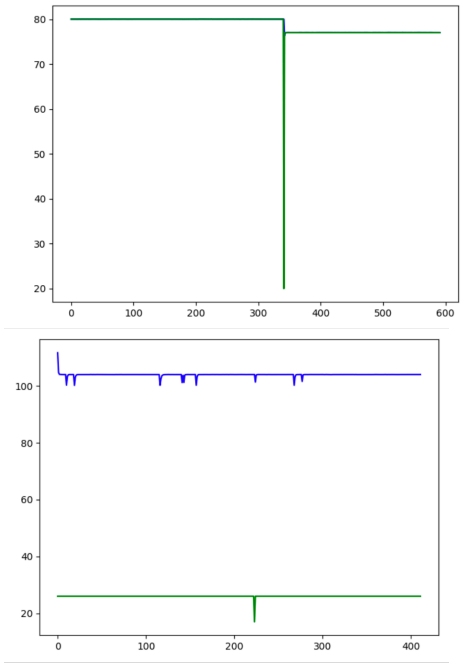
V. Results

The algorithm performed quite well before training and showed improvement after some training.



Learned Parameters: The green line represents the true bpm of the entire *Goldberg Variations* as encoded in the MIDI files and in blue is the estimates of the bpm based on the results of the parameters in our Kalman filter after gradient ascent. On the *y* axis is the bpm, on the *x* axis is time in seconds. The mean square error of our estimate is 92.2

It is not unlikely that the gradient ascent converged on a local maximum and that we could find even better parameters. That being said it performed extremely well on the simplest of pieces such as the *Prelude* to Bach's first *Cello Suite*, in which there is almost no rhythmic variation between musical events and even still well on the *Adagio* from Bach's first *Sonata for Solo Violin* which is rhythmically varied.



Above, *Prelude*, Below, *Adagio*: In the simple *Prelude* from Bach's first *Cello Suite*, the blue line representing our estimate is nearly invisible below the true bpm in the MIDI file. The MSE is 0.0009.

In the *Adagio* from Bach's first *Sonata for Solo Violin*, there are long pauses without musical events and the notes shift quickly between long held notes quarter notes and short quick thirty-second notes. The algorithm guessed a bpm that was 4 times the true bpm in the MIDI file but still I calculated an MSE 0.0638 because I feel that finding a bpm that is a double or quadruple of the actual bpm is still a good performance. Also, a bpm as low as the true bpm could possibly be perceived by the listener as being higher than it is by double or quadruple.

VI. Conclusion and Future Work

The Kalman filter has proven a very effective tool for updating the belief about the beat in music. The beat finding algorithm performed well and improved after training though intuition suggests that extensions to the local search algorithm may provide further improvements. Random starting points and genetic algorithms may be the ticket. A larger training set may also help. Further knowledge about the pitch of the musical events can be used to weight each musical event during each

observation. This would require including a distribution over pitch in the belief state.

Additionally, the existing algorithm can be improved by introducing additional training data. Bach is regarded as a master of rhythmic and harmonic counterpoint and his musical influence has left an imprint on much of western music. However, many modernist twentieth century composers have composed works that subvert convention and while their works still contain beats, the model for how the notes follow these beats and the expectation of a steady beat may be drastically different from their classical predecessors.

Further work to extend the algorithm to be more useful would be to incorporate online musical event detection from an audio source. This would allow the algorithm to determine the beat of live music. Applications of such an algorithm would be useful for building creative tools for musicians and composers i.e. an application for the composer that could play along or embellish existing music or for the musician that could sync up a digital score to live music so that they could follow along.

ACKNOWLEDGMENT

A big shoutout to the author of article[4] for helping the author understand the intuitions behind the Kalman filter, which helped guide this paper and its process.

Another enormous shoutout to Dave Grossman's impeccable MIDI transcriptions of Bach's seminal works. What an achievement that is. Dave is a legend. And he wins the award for the most beautiful website to come out of the 90s.

And thank you Bach for always being an inspiration.

Lastly, a shoutout to the teaching staff of AA228. Thank you so much.

REFERENCES

[1] Eric D. Scheirer, "Tempo and beat analysis of acoustic musical signals," J. Acoust. Soc. Am. Vol. 103, No. 1, 1998, 558-601. [5] Special issue on Wavelets

[2] George Tzanetakis, Georg Essl, Perry Cook, "Audio Analysis using the Discrete Wavelet Transform," Proceedings of the WSES International Conference Acoustics and Music: Theory and Applications (AMTA 2001)

[3] Mykel J Kochenderfer. Decision making under uncertainty: theory and application. MIT press, 2015.

[4] Tim Babb. How a Kalman filter works, in pictures. Aug. 2015. url: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

[5] Johann

[6] David J. Grossman. How a Kalman filter works, in pictures. Aug. 2015. url:

[7] Tim Babb. Dave's J.S. Bach Page: MIDI Files. Oct. 1997. url: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/