Modeling the Induced Physiological Response to Local Musical Features via Input-Output Hidden Markov Models

David Rector

Summary

This "final" paper will serve as a status report on a longer-term project than initially planned: the implementation of an ambitious model (IOHMMs) to a complex data set (biometric measures of physiological response to local musical features). An overview of the model and methods will be followed by an examination of some preliminary results and issues encountered during implementation. A final discussion will explore next steps.

I. Model

Input-Output Hidden Markov Models were proposed in 1995 [1] as a means of applying feature-based machine learning techniques to sequential data while maintaining the advantages of modeling such data as consisting of Markov chains – in other words, capturing the concept of target observations being influenced by *both* input features *and* by its previous "state," the latter providing "context" with which to interpret the input.

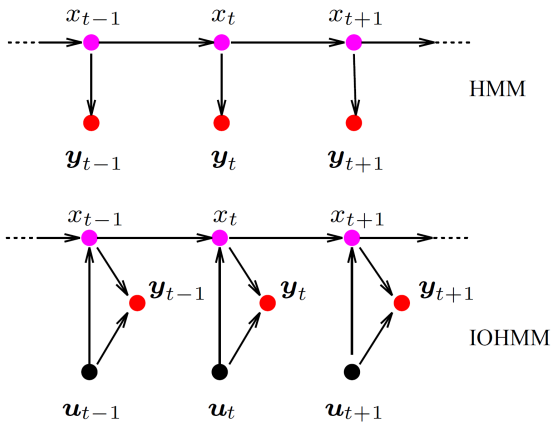

Figure 1 [1]. *A comparison of the Bayesian dependency structure of HMMs and IOHMMs. A standard HMM (top) models targets (y) as influenced only by their current state, which in turn is influenced only by its previous state. The IOHMM (bottom) allows for the output to be additionally influenced by conditionally independent input features (u).*

The term "Input-Output" refers to the notion that the model can be used to translate input sequences into output sequences, though two things should be noted: a) the input and output sequences need not be the same length or occur in synchrony [3], and b) the model can be used equally well to map input sequences to single, non-sequential outputs (e.g. "sequence tagging" classifiers [5]), or map single input feature vectors to output sequences. IOHMM's are not the only possible means of inserting feature information into sequence prediction (e.g. Maximum Entropy Markov Models – MEMM's – combine feature input with Markovian assumptions and have been used successfully for sequence tagging [6]), but their ability to map a *continuous* input sequence to a *continuous* output sequence, and the flexibility permitted by the model's modular structure (discussed in a moment), attracted the researcher to them for the present task: correlating a sequence of input feature vectors extracted from music to the sequence(s) of a listener's physiological response data, captured by biometric sensors, that the music may have induced in him.

The model can be constructed as follows. The researcher first chooses a finite number of states that he chooses to model as latent in the data. For example, when an experimental subject is disinterested or distracted, he will respond to input data differently than when he is focused – it can be thus be said that observations emitted while in the "disinterested" state will relate to the input differently than observations emitted in the "focused" state. If the researcher wanted to make only this single distinction, he might use a model consisting of only two states.

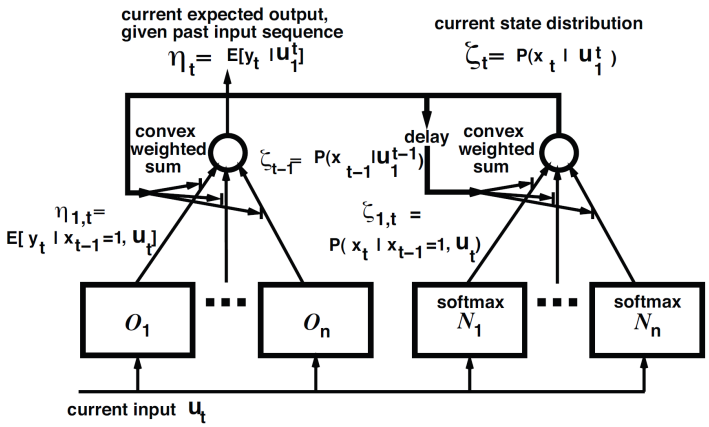

Figure 2 [1]. *The architecture of the IOHMM.*

Each state is then represented in the performance of two distinct tasks: a) predicting the *next* state probability distribution given the present state probability distribution and the current input features, and b) predicting the *current output* given any particular state. Each state will thus be provided two "subnetworks," one to perform each of these tasks. The state prediction task will plainly influence the output prediction task as, essentially, a state choosing function in a "mixture of experts" model. More specifically, at each time or step in the sequence, the output prediction subnetworks each develop their own prediction of the next output distribution given the current input feature vector (for a continuous Gaussian output, this might be the predicted mu and sigma for each degree of freedom of output), and the state prediction subnetworks combine their predictions with a variable (ζ_t) representing the combined previous predictions to form an overall next-state probability distribution. This distribution then weights the probability distributions of the output prediction subnetworks by the individual probability of their associated states, and sets the overall output of the model the expected value of the weighted-sum distribution. (A trivial alteration to this step in the model might here be proposed: *discretely* choosing - based on the expected value of the state probability distribution - the *single* output prediction subnetwork whose prediction will become the final predicted output; this might increase performance for some tasks, though certainly not for all.) The probability distribution of the state probability subnetworks is then, of course, used to weight the next set of predictions from the subnetworks.

In this way, the variable ζ_t contains what might be said to be the "memory" of the system at time t. Because the functioning of this "memory" is critical to the model's ability to emulate systems with more complex memory – like humans listening to music – we will consider it closely in section four ("Next Steps").

The "modular" structure of IOHMMs alluded to above is this: the state prediction and output prediction subnetworks need not be mere linear/quadratic/etc. functions of the input features; instead, they can in and of themselves be any complex machine learning implementation that maps inputs to outputs using some set of distinct parameters or weights whose values can be learned through training. Bengio and Frasconi proposed Artifical Neural Networks, and indeed this researcher attempted to implement this proposal, as noted later. But one can conceive of the potential of inserting *different types* of modules in each state's subnetworks, depending on the complexity or nature of the predictions that might be required in that state.

The modeling potential of the Input-Output Hidden Markov Model is perhaps best illustrated in a 2005 paper using IOHMMs to forecast daily electricity prices in the Spanish market [4]. As seen in Figure 3, electricity prices offer a good test case for IOHMMs – different market states are plainly distinguishable; furthermore they may be explained by certain input features, among which the authors chose past hourly production by various energy-production technology in use, the hourly system demand, and a variable accounting for different lags of the price. The authors chose to model four states as an optimum that would offer considerable explanatory power while minimizing computation time and over-fitting. The model performed ably at forecasting prices, specifically by demonstrating an ability to distinguish states using complex considerations of the input features:

It is important to note that in this context, market states are not related to price levels but rather to a functional relationship between the set of input variables and the marginal price. This fact can be observed in Fig. 8 *[Figure 3 here]*. This figure shows a zoom of price series from April to July. Although, during May, minimum prices were around 1.5 c and maximum prices close to 3.5 c Euros, as in the second period, the series has already recovered the stability and weekly rhythm. The model is able to capture this switch, and therefore, May is classified in the third period. [4]

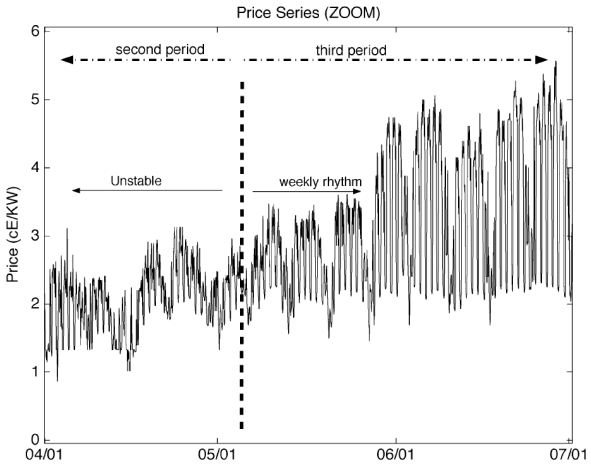

Figure 3 [4]. *Changing market states – one chaotic, one featuring stable price rhythms – can be discerned in the Spanish electricity market, and predicted by IOHMMs.*

The question this researcher considered was how IOHMMs might perform in the task of classifying short-scale, high variance sequences that surely involve complex mappings of the input: predicting human physiological response to local musical features.

II. Methods

This researcher has gone into detail regarding his data collection methods in a previous milestone report; he will briefly summarize them here, then devote considerable space to detailing his modeling and training methods.

Data collection

The researcher used himself as the sole experimental subject for this preliminary project; he understood such results would perhaps not be generalizable but he wanted to ensure he could gather data he could trust. He obtained a Neurosky EEG headset and found its proprietary "Esense" measures to correlate with own subjective sense of physiological arousal or relaxation. He additionally obtained a light-based pulse meter, capable of providing data on instantaneous heart rate, pulse volume at the point of measurement (a fingertip), and heart rate variability. He constructed a rudimentary (though reliable) skin conductance circuit.

He substantially altered existing Java code created by Eric Blue [7] for the aforementioned EEG headset to combine Raw EEG (recorded at 500 Hz), the Esense measures mentioned above (output every second), and pulse and skin conductance information into CSV files. He created a computer-generated piano performance with high verisimilitude of all 32 of Johann Sebastian Bach's "Goldberg Variations" using MIDI sequences obtained from the internet [8]. In brief, these were chosen for two reasons. First, there are relatively few degrees of freedom in musical features that cannot be captured by analysis of MIDI files – for example, since the Goldberg Variations were written for harpsichord, an instrument with no dynamic variability, they remain musically interesting when played with a very limited range of key velocity, as transcribed in this instance – thus reducing the possible confounding factor of physiological response induced simply by great dynamic variability. Secondly, as a theme and variations, they permit relatively (though not entirely) controlled comparisons of the effect of certain musical features – i.e. the chord progression might be identical in the corresponding spots of two variations, but note density, melodic contour, rhythm, and tempo might be very different.

The researcher extracted MIDI features using the Matlab-based "MIDI Toolbox" [9]. He chose to represent input features and output sequence synchonously, every one second. Initial features chosen for analysis included four simple and one complex feature: the melodic range of notes sounding during the past second, the density of note onsets over the last second, the density of note onsets not occuring concurrently (within a small threshold), and the average "tonal stability" for note onsets during that second [10]. He included the possibility to (and in the future will) use various audio features (e.g. spectral roughness) extracted from the computer-generated WAV file; these will permit consideration of measures of consonance or dissonance not feasible to capture from MIDI representations. For now he limited his analysis to these basic MIDI features.

Model construction and training method

He chose to model three states initially, though this will be a parameter in need of adjustment as the project proceeds. He chose, initially, to simply map input features to a single output sequence – the listener's heart rate, as he determined after several testing sessions that, for him, change in heart rate was a relatively repeatable listening phenomenon to a given piece of music.

He chose to use, in line with Bengio and Frasconi's proposal [2], Artificial Neural Networks as all output prediction and state prediction subnetworks, each with five layers and trainable using Levenberg-Marquardt (LM) backpropagation due to this algorithm's relatively high computation speed. The parameters to be optimized were thus the weighting coefficient matrix of each of the neural networks – a total quantity of 2 times the number of states, multiplied by the number of layers, multiplied by the number of features – 2 * 3 * 5 * 5 = 150 parameters. The model, coded in Matlab, followed closely the broad outline given in [2]. Optimization of the model would be attempted through a type of generalized expectation maximization – the goal, at each iteration, being the following:

1. ESTIMATION. Given the current parameters, calculate:
    a) The predicted state transition probability matrix – that is, a matrix of size [(numberOfStates) x (numberOfStates) x (sequenceLength)], giving the probability distribution, as calculated by the state transition prediction subnetworks, at each step of the input sequence.
    b) The expected outputs matrix – that is, a matrix of size [(2 * outputDegreesOf-Freedom) x (numberOfStates) x (sequenceLength)], for a continuous Gaussian output, giving the predicted mean and standard deviation of each output prediction subnetwork at each step of the input sequence.
    c) The transition posterior probabilities – that is, the conditional probability, given the input, of a transition from each possible current state to each possible successor state at each step of the input sequence.
    d) The state posterior probabilites – that is, the conditional probability, given the input, of being in any particular state at each step of the input sequence – in other words, the sum of the transition posterior probablities over their current states.
    e) The probability of the observed target output given the input features and each possible state – that is, for a continuous Gaussian output, the distance of the target vector relative to each output subnetwork's predicted mean and standard deviation.
2. MAXIMIZATION. Adjust the parameters of each subnetwork (and any other parameters) to maximize the likelihood of the data given the parameters. This was performed as follows:
    - Maximize relative to the state transition probability matrix:

- Compute the partial derivatives at each value of the state transition probability matrix with respect to each possible parameter of the state transition prediction subnetworks – that is, change each value of the state transition probability matrix by a certain step size (chosen as a constant, 0.05, for now, later to be made "greedy" at first and decreasing in size as likelihood increases); re-normalize so that the transition probabilities for that particular current state sum to one; train a copy of each state transition subnetwork using LM back-propagation on the new possible state transition probability matrix; and calculate how the parameters/weights of each subnetwork changed with respect to the change of the current value of the state transition probability matrix. (Note that the transition matrix was initialized as uniformly 1/3 - all transitions were equally probable at all times.
- Use the above partial derivatives and the state transition posterior probabilities (1c above) to calculate the overall partial derivative of the likelihood of the data given the parameters with respect to each of the state transition prediction subnetworks' 75 possible parameters. [2]
- Update the weights/parameters of the state transition subnetworks by adding these 75 partial derivatives to the current values of those 75 parameters/weights – this will, in effect, boost the value of parameters whose increase causes an increase in the likelihood of the data and decrease the value of parameters whose increase would cause a decrease in the likelihood.

◦ Maximize relative to the expected outputs matrix in a similar fashion:

- Compute the partial derivatives of each value of the expected outputs matrix with respect to each of the 75 possible parameters of the output prediction subnetworks. (The step size was made to be 1.0 for both the mean and variance in the expected outputs matrix – again, this will be made adjustable later. Additionaly note that each output prediction subnetwork was initially trained identically using the full sequence of input features and the full sequence of output features.)
- Additionally compute the partial derivative of the logarithm of the probability of the observed target output (analogous to 1e above) with respect to the change (i.e. step size) in expected output.
- Use the above two partial derivatives and the state posterior probabilities (1d) to calculate the overall partial derivative of the likelihood of the data given the parameters with respect to each of the 75 output prediction subnetwork's pararmeters. [2]
- Add these the weights/parameters to the current values of the output prediction subnetworks in a similar fashion as above.

3. Having updated the parameters/weights of the subnetworks, check to see if the overall rate of change of the likelihood given the new set of parameters is greater than a certain very low threshold, or if the maximum number of iterations has been reached – if so, this will be the final parameter set; if not, go back to step 1 and start over with the updated parameters.

III. Implementation issues

The researcher's time spent collecting data left him with less time for debugging the complex model he constructed; thus, optimization has not thus resulted in expected optimum values. More specifically, the researcher concedes little faith that his implementation of the maximization portion of the EM algorithm contains no errors, and certainly no faith that it it the fastest possible method, which is the more critical issue – even with few states and few layers, 100 seconds (i.e. sequence length of 100) of sequences requires 10-15 minutes on a quad-core Intel Xeon 3.0 GHz processor to compute anything close to an optimum set of parameters; and worse, the model often fails to correctly converge upon the optimum. Lastly, the partial derivatives the researcher computes often result in NaNs; the researcher is uncertain whether this is due to computational underflow correctable by using logarithms or a different bug altogether. Plainly there remain bugs and the researcher continues to work to correct them. He welcomes advice on solving these issues.

In addition, there are issues that might be hidden from sight amidst the larger ones. For one, the model was far more prone to update the expected output matrix than the transition probability matrix, the latter of which remained at the values at which it was initialized: a constant 1/3 for every value. This could be the result of two issues (disregarding for the moment possible bugs in the code): first, the relative step sizes of the output prediction and state transition prediction subnetworks may need adjustment (or, better yet, made to automatically adjust relative to the overall rate of change of the likelihood function relative to their half of the set of parameters); second, the limited computation speed of the algorithm has limited the possible sequence length to such a degree that broad state transitions are hardly possible even when considering their "true," latent values, much less when considering noisy data that may or may not reflect them.

A second, possibly insidious issue: the EM algorithm appears to exhibit some degree of "oscillation" between maximizing the likelihood with respect to the state transition prediction subnetworks' parameters and the output prediction subnetworks' parameters. Perhaps, after correcting some of the current issues, this would become non-relevant, and these oscillations would "dampen" with each iteration to allow convergence, or disappear altogether. If not, it may be too "greedy" to update both the state transition prediction subnetworks' parameters and the output prediction subnetworks' parameters together on each iteration – there may need to be a second degree of alternation, in addition to that between Estimation and Maximization, between maximization of each of these two sets of parameters.

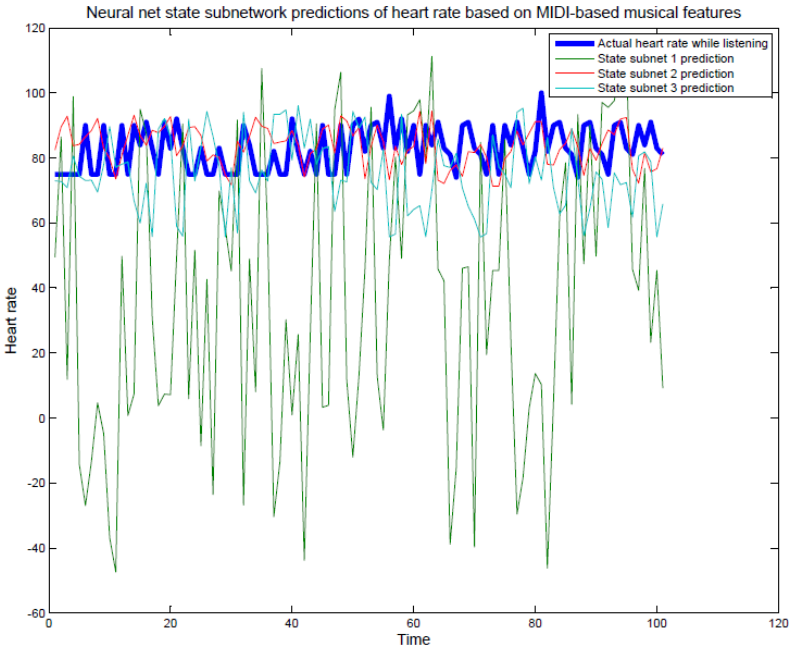

Figure 4. *Predicted mu values of each of three output prediction neural network-based subnetworks, compared to actual heart rate.*

Given these issues, and in particular the lack of response of the state transition probability matrix to parameter changes of the state transition subnetworks, the researcher has not yet implemented the "memory" function used to determine a final predicted output by choosing or weighting the predictions of the output prediction subnetworks based on the predictions of the state transition subnetworks. However, he has some reason for optimism, as a simply glance at the output prediction subnetworks' predicted outputs shows promise for the model's ability to divide the task of predicting outputs to different state subnetworks; as one can see in Figure 4, individual subnetworks seem to occasionally be able to "pick out" spikes in arousal, while being wildy incorrect at other times. Were the state prediction subnetworks able to correctly choose which state's prediction to use at each time, the model may have considerable promise. The researcher concedes the possibility of over-fitting or an overly optimistic analysis of the graph.

IV. Next steps

After correcting the plain algorithmic/computational issues detailed above, the researcher's second steps involve speculation to address more nuanced, theoretical issues.

First, it will be critical that the researcher implements a means of escaping local maxima in the likelihood maximization step – a "random restart" implementation is probably the most advisable. Furthermore, any simplifications that might speed up maximization time – for example, accepting the first result that results in the slightest maximization of the likelihood, rather than proceeding through every possibility. The possibility of somehow simplifying the data into a classification problem, so as to potentially take advantage of a much faster basic EM solution, is also worth considering, if it does not limit the explanatory power of the model.

The most interesting issue that the researcher has considered involves the memory variable ζ_t. Under Markovian assumptions, it is said to contain the entire context/"memory" of all the states of prior to time t. Theoretically it makes formulae clean; computationally it makes code efficient, and to be sure, this assumption has indeed proven remarkably accurate and useful in many applications involving human data (e.g. speech recognition). Can it accurately represent a human's memory as it relates to music?

The Goldberg variations shall provide the perfect test case when the researcher can get the algorithm working properly. The famous "Aria" begins the work. The motivic material of the Aria is then transformed in every conceivable way into thirty variations that evoke states ranging from ecstasy to deep contemplation. Then, finally, the "Aria da capo" - the return of the Aria; identical notes, but somehow, a profoundly different character. Joseph Campbell describes in "The Hero's Journey" the universal human mythology of the hero who ventures away from home, encounters dangers and exploits that test his character, and returns a different person, having attained not quite what he expected, but nothing less. It is in this sense that a listener perceives the Aria da Capo – it is a different Aria than that which began the piece. Can ζ_t, the memory vector, giving simply the current probability distribution of the listener's latent "states" given merely the current input features and ζ_{t-1}, capture this changed relationship of the listener to the input features?

Obviously this is absurd; this kind of complex relationship-based "memory" is not what Markovian assumptions intend to model. However, the researcher wonders if alterations that could bring this variable more in line with human memory function are not so far fetched. Consider first the present formulation: training the model on simply the biometric data obtained while listening to the first Aria and then, sixty minutes later, the Aria da Capo, will simply "confuse" it – it will receive identical input features but far different output targets (or rather, far different predicted "states"); the model will thus

interpret the Aria's input features as simply producing very high variance, and will predict this high variance for both the beginning Aria and the Aria da Capo. Certainly one could hazard that the burden of properly distinguishing the two inputs must be placed more on the input feature set than on this memory variable; that this variable cannot in fact contain any truly complex "memory" in the sense humans perceive it, that some of this memory must be submitted via a music theoretical analysis provided in the input features. But consider if, simplistically, an accounting for *repetition* of input features (and, perhaps, of state probability distributions?) could be included in the calculation of the memory vector, or if more distant dependencies of ζ_t than ζ_{t-1} could be included; a careful construction of such dependencies may well bring this variable more into line with human memory function. What's more, given the current complexity of the model, introducing additional dependencies into ζ_t wouldn't seem to affect computational speed to the extent that other factors currently affect it. The researcher will continue to ponder the problem.

References

[1] Bengio, Yoshua, and Paolo Frasconi. "An input output HMM architecture." *Advances in neural information processing systems* (1995): 427-434.

[2] Bengio, Yoshua, and Paolo Frasconi. "Input-output HMMs for sequence processing." *Neural Networks, IEEE Transactions on* 7.5 (1996): 1231-1249.

[3] Bengio, Samy, and Yoshua Bengio. "An EM algorithm for asynchronous input/output hidden Markov models." *International Conference On Neural Information Processing*. Hong-Kong, 1996.

[4] González, Alicia Mateo, A. M. S. Roque, and Javier García-González. "Modeling and forecasting electricity prices with input/output hidden Markov models." *Power Systems, IEEE Transactions on* 20.1 (2005): 13-24.

[5] Marcel, Sebastien, et al. "Hand gesture recognition using input-output hidden Markov models." *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. IEEE, 2000.

[6] McCallum, Andrew; Freitag, Dayne; Pereira, Fernando (2000). "Maximum Entropy Markov Models for Information Extraction and Segmentation". *Proc. ICML 2000*. pp. 591–598.

[7] Blue, Eric. (2011) *Neurosky EEG Data Streamer* [Computer program]. Available at http://eric-blue.com/2011/07/24/mindstream-neurosky-eeg-data-streamer/.

[8] *Goldberg Variations*. Bach, J.S. MIDI Sequences by Bruno De Giusti. Available at http://www.kunstderfuge.com/bach/harpsi.htm#Arias%20and%20Variations

[9] Toiviainen, Petri and Eerola, Tuomas. (2006) *Midi Toolbox* [Matlab functions]. Available at https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/miditoolbox.

[10] Krumhansl, C. L. (1990). Cognitive Foundations of Musical Pitch. New York: Oxford University Press.