Hierarchical Linear Dynamical System for Representing Notes from Recorded Audio

Leila Kalantari  
University of Florida  
Gainesville, USA  
leila@ufl.edu

Jose Principe  
University of Florida  
Gainesville, USA  
principe@cnl.ufl.edu

Kathryn E. Sieving  
University of Florida  
Gainesville, USA  
chucao@ufl.edu

Abstract—We seek to develop simultaneous segmentation and classification of notes from audio recordings in presence of outliers. The selected architecture for modeling time series is hierarchical linear dynamical system (HLDS). We propose a novel method for its parameter setting. HLDS can potentially be employed in two ways: 1) simultaneous segmentation and clustering for exploring data, i.e. finding unknown notes, 2) simultaneous segmentation and classification of audio recording for finding the notes of interest in the presence of outliers. We adapted HLDS for the second purpose since it is an easier task and still a challenging problem, e.g. in the field of bioacoustics. Each test clip has the same notes (but different instances) as of the training clip and also contain outlier notes. At test, it is automatically decided to which class of interest a note belongs to if any. Two applications of this work are to the fields of bioacoustics for detection of animal sounds in audio field recordings and also to musicology. Experiments have been conducted for segmentation and classification of both avian and musical notes from recorded audio.

Index Terms—Time series classification, time series clustering, anomaly detection, outlier detection, Kalman filtering, kernel Mahalanobis distance, classification, adversarial robustness, time series segmentation

I. INTRODUCTION

High efficiency monitoring of biodiversity and ecosystem health is a priority in wildlife ecology. Many wildlife species (especially birds) are highly detectable by their sounds, therefore passive acoustic monitoring is appropriate to detect and survey such species. Yet, without accurate computer-automated retrieval of acoustic data, terabytes of recordings are useless. Our work can facilitate data retrieval strategies relying on signal processing and machine learning by helping eliminate the needs for manual recalibration, detector pre-training, and coping with variable acoustic conditions in the recording environment.

II. TECHNICAL BACKGROUND

A. Kalman filtering

Consider the following equations to model time series \{y_t\}:

\[
\begin{align*}
x_{t} &= F_{t-1}x_{t-1} + \epsilon_{t}^{x} & \text{Hidden layer} \quad (1) \\
y_{t} &= H_{t}x_{t} + \epsilon_{t}^{y} & \text{Observation layer} \quad (2)
\end{align*}
\]

Integer \(t\) is the time index, vector \(x_t\) is the (hidden) state, vector \(y_t\) is the observation, \(F_t\) is the transition matrix, \(H_t\) is the observation matrix, \(\epsilon_{t}^{x} \sim \mathcal{N}(0, R^{x})\) is the uncertainty vector for the hidden layer and vector \(\epsilon_{t}^{y} \sim \mathcal{N}(0, R^{y})\) is the measurement error.

Kalman filtering efficiently calculates \(\hat{x}_t\), an estimation of the current state \(x_t\), given the estimation of the previous state \(\hat{x}_{t-1}\), previous information matrix \(P_{t-1|t-1}\), and the current observation \(y_t\), in an online fashion assuming the measurement and transition matrices are known for each time step. Let \(I\) denote the identity matrix and set the initial state and the initial information matrix as \(x_0 = (H_0^T H_0)^{-1}H_0^T y_0\) and \(P_{0|0} = (H_0^T H_0)^{-1}\) respectively. One step of Kalman filtering is:

\[
\begin{align*}
&1: \text{function KALMAN-STEP}(y_t, \hat{x}_{t-1}, P_{t-1|t-1}; H_t, F_{t-1}) \\
&2: \quad \hat{x}_{t|t-1} \leftarrow F_{t-1} \hat{x}_{t-1} & \triangleright \text{A priori state estimate} \\
&3: \quad P_{t|t-1} \leftarrow F_{t-1} P_{t-1|t-1} F_{t-1}^T + R^{x} & \triangleright \text{A priori info matrix} \\
&4: \quad G_{t} \leftarrow P_{t|t-1} H_{t}^T (H_{t} P_{t|t-1} H_{t}^T + R^{y})^{-1} & \triangleright \text{Kalman gain} \\
&5: \quad \hat{x}_{t} \leftarrow \hat{x}_{t|t-1} + G_{t}(y_t - H_t \hat{x}_{t|t-1}) & \triangleright \text{State update} \\
&6: \quad P_{t|t} \leftarrow (I - G_{t}H_{t}) P_{t|t-1} & \triangleright \text{Info matrix update} \\
&7: \quad \text{return } \hat{x}_t, P_{t|t} \\
&8: \text{end function}
\end{align*}
\]

B. HLDS

HLDS models time series using a set of hierarchical linear systems [1]. Without loss of generality and for ease of presentation, assume \(L = 2\) where \(L\) denotes the number of hidden layers. The system of equations is:

\[
\begin{align*}
z_{t} &= z_{t-1} + \epsilon_{t}^{z} & \text{2nd hidden layer} \quad (3) \\
x_{t} &= F_{t}x_{t-1} + Bz_{t-1} + \epsilon_{t}^{x} & \text{1st hidden layer} \quad (4) \\
y_{t} &= H_{t}x_{t} + \epsilon_{t}^{y} & \text{Observation layer} \quad (5)
\end{align*}
\]

where \(H\) is the observation matrix, \(F\) is the transition matrix, \(B\) is the coupling matrix, \(\epsilon_{t}^{x} \sim \mathcal{N}(0, r^{x} I)\) is the measurement error, \(\epsilon_{t}^{z} \sim \mathcal{N}(0, r^{z} I)\) is the uncertainty term for the second hidden layer, \(\epsilon_{t}^{y} \sim \mathcal{N}(0, r^{y} I)\) is the uncertainty term for the first hidden layer and \(r^{x} I, r^{y} I, r^{z} I\) are covariances for the corresponding Gaussian distributions. Assuming isotropic noise model for each layer is a simplification compared to the model described by equations (1) and (2) requiring less hyperparameters to be set (Section II-B).

It is crucial that the transition matrix of the layer associated with \(z\) is identity. According to the model, time series \{\(z_t\)\}
is unlikely to take a big leap over one time step if \( r^x \) is set small. This self-organizing property allows utilizing HLDS for the purpose of clustering.

Rewrite equations (3)-(5) into canonical form as:

\[
\begin{align*}
\dot{x}_t &= \tilde{F} \tilde{x}_{t-1} + \tilde{e}_t & \text{Joint hidden layer} \\
y_t &= \tilde{H} \tilde{x}_t + \tilde{e}_t^y & \text{Joint observation layer}
\end{align*}
\]

where \( \tilde{x}_t = \begin{pmatrix} x_t \\ \tilde{x}_t \end{pmatrix} \) is the joint hidden state, \( \tilde{F} = \begin{pmatrix} I & 0 \\ B & F \end{pmatrix} \) is the joint transition matrix, \( \tilde{H} = \begin{pmatrix} 0 & H \end{pmatrix} \) is the joint observation matrix and \( \tilde{e}_t = \begin{pmatrix} e_t^x \\ e_t^y \end{pmatrix} \) is the joint uncertainty vector.

### III. The Proposed Method

We use the HLDS architecture from [1]. HLDS imposes a constraint that effectively regularizes the Kalman specifically for note segmentation. The time series data is modeled as a process that has multiple layers of hidden states. Higher layers have lower dimensions compared to lower ones. Also, higher levels exhibit longer term stationarity.

In this paper, we propose to achieve the relative stationarity of layers in two novel ways: 1) Each layer is some low pass filtering of two consecutive states from the level just below it plus innovation (Section III-A). 2) Relative stationarity is also controlled by setting the innovation variances of higher layers smaller compared to lower ones (Section III-B). The specific block structure of the joint transition matrix pioneered by [1] provides efficient online updates (using Kalman filtering) to the hierarchy of hidden states of the process.

#### A. Parameter setting

In HLDS, parameters are transition and coupling matrices for each hidden layer. We propose to set them explicitly instead of training for them to avoid overfitting specially when training data is limited. Parameters of HLDS are set in a way that each hidden layer acts as low pass filtering of two consecutive (in time) states of the previous hidden layer (plus innovation modeled by Gaussian distribution). Let \( N \) and \( S \) be the dimensions of \( x_t \) and \( z_t \) respectively and \( N \) be divisible by \( S \). The model is:

\[
\begin{align*}
z_t &= z_{t-1} + \epsilon_t^z & \text{2nd hidden layer} \\
x_t &= -x_{t-1} + B z_{t-1} + \epsilon_t^x & \text{1st hidden layer} \\
y_t &= x_t + \epsilon_t^y & \text{Observation layer}
\end{align*}
\]

where \( B = \left( B_1, \ldots, B_S \right)^T \). For each \( s \in \{1, \ldots, S\} \), block \( B_s \) is an \( N/S \times S \) matrix whose \( s \)th column is \( (2S/N)1 \) and zero elsewhere. The set of equations above can be rewritten in canonical form as in equations (6)-(7) to be able to exploit Kalman filtering.

#### B. HyperParameters setting

1) Innovation and error parameters: Other than our proposed parameter setting (Section III-A), the way we set innovation parameters contributes significantly to the relative stationarity of higher layer hidden states compared to lower ones. There are \( L \) Gaussian parameters to be set where \( L \) is the number of the hidden layers. They are denoted by \( r^z \) and \( r^x \) when \( L = 2 \). These hyperparameters should relate to each other in a way that none of the corresponding terms in \( \lambda(x_t) = (y_t - H x_t)^T (R^y)^{-1} (y_t - H x_t) + (x_t - \tilde{x}_{t-1}^T P_t^{-1} (x_t - \tilde{x}_{t-1}^T) \) overwhelms another (the cost function KALMAN-STEP implicitly solves for). We achieved this by setting the innovation parameter of each layer proportional to the dimension of the layer. We set \( r^y \), the error parameter, equal to be the innovation parameter of the bottom hidden layer in all our experiments. We are aware that the best choice depends on the data. However, we are able to achieve successful results in both of our experiments without considering the data (Sections IV and V).

2) Number of hidden layers: Another hyperparameter is \( L \), the number of hidden layers. In our experiment, we set \( L \) to 4. In future, we will study whether having more or less layers has any merits. The minimum number of hidden layers necessary for the architecture of HLDS is two.

3) Sliding time domain window length: It is desirable to decrease \( w \), the length of the sliding window, to improve the computational performance. However, \( w \) should not be set too small. It has to be set large enough to contain enough information in a clip and small enough to maintain the stationarity assumption.

C. Training and testing

In the training step, observations \( \{y_t\} \) from each training note are mapped to the \( z \)-space in the self-organizing manner implicated by HLDS model. Note that “training” notes are only utilized for this step and parameters are set explicitly (Section III-A). These mappings are to be compared to where test observations get mapped to. We still utilize Kalman filtering during training and testing to calculate states \( \tilde{x}_t \), associated with each train or test observation, in order to extract \( \{z_t\} \) as in [1]. Mappings \( \{z_t\} \) can be exploited in many ways (next section).

D. Post-processing

If \( z \)-mappings from a training note form a single convex isotropic cluster, a simple distance-to-the-cluster-center heuristic in \( z \)-space can be applied at the test time for the purpose of classification. If the corresponding clusters have more complicated shapes as with avian notes, a secondary classification algorithm can be applied. If the system is to be tested with outliers at the test time, a generative classification model can be applied in the \( z \)-space [2], [3]. We applied kernel Mahalanobis distance for possibilistic classification [4] for two reasons: It can discern outliers at the test time, and it has fully automated parameter setting. Therefore, HLDS is the feature representation component in our prototypical classification system with PC-KMD as another component.

### IV. Experimenting with avian notes

The dataset of avian notes as used in our experiment is described here. Carolina chickadee (Poecile carolinensis) and
tufted titmouse (*Baeolophus bicolor*) are the two avian species are abbreviated by *pocaro* and *babico* throughout the paper. Various notes uttered by *pocaro* and *babico* were recorded at 1 m distance away from the bids. The available labeled notes are *babico*-D, *babico*-F, *pocaro*-Z, *babico*-A, *babico*-Z, *pocaro*-C, *pocaro*-D and *pocaro*-E. The number of instances of each note is 15, 6, 4, 4, 6, 7 and 10 respectively for a total of 50 instances.

The training clip consists of two instances of each of the *babico*-D, *babico*-F, and *pocaro*-Z (Figure IV(a)). The test clip consists of the rest of the labeled data in a random order. Matrix *H* is set to identity and the time domain sliding windows \{y\}_t are preprocessed as \(y'_t = |\text{det}(y'_t)|\) before applying HLDS. Let \(w'\) and \(w\) denote the dimensions of \(y'_t\) and \(y_t\) respectively. The hyperparameters of HLDS are set as follows: the dimension of hidden layers are \([96, 24, 12, 2]\) which implies \(w' = w = 96\) since \(H\) is set to identity; \(q = 48\) where \(q\) denotes the number of overlapping samples between two consecutive sliding windows. Each of parameters \(r^y, r^x, \ldots, r^x\) is set proportional to the dimension of its corresponding layer as described in Section III-B. We emphasize that all hyperparameter choices are confirmed by getting reasonable results on training data (as illustrated in Figure IV(b)-(d)) without considering the results on test data (Table I). One direction of future research is automation of hyperparameter setting. During the training phase, we utilize the labeled notes to find clusters which are illustrated in Figure IV(b). Based on comparing test z-representations to training z-representations, we assign 3 soft scores to \(y_t\) where each score represents \(y_t\)’s degree of membership to the corresponding training class (Figure IV(c)). For this last step, we use a fusion of OCGPs (Section III-D). To make crisp decisions from soft scores of OCGPs, threshold \(\tau = 6\) is used. All predictions below \(\tau\) were discarded. Then all predictions that lasted for less than 20 time steps were discarded since a note cannot last that short. Note that these last hyperparameters (number of time steps = 20 and \(\tau = 6\)) are hyperparameters of the system, not of HLDS. This is a simple example of how expert knowledge can be used to make crisp decisions from soft classification scores. An example of final class assignments is illustrated in Figure IV(d). The resulting test confusion matrix is illustrated in Table II. We wish all instances of the training classes be assigned their true labels at the test time and the rest of the notes to be detected as outliers. *babico*-D notes are often mispredicted in bioacoustics softwares because of their broad band even in noiseless recordings. The perfect prediction by our system is encouraging. Note that *babico*-F is not confused with *babico*-D. Bioacoustics softwares have trouble distinguishing them. Species *babico* and *pocaro* should be indistinguishable wrt their Z notes because across species this important signal (of imminent attack by a hawk) is highly canallized. Therefore, the confusion between *babico*-Z and *pocaro*-Z, observed in our results, is desired. The misprediction of *pocaro*-D notes can potentially be prevented at different stages of our system.

Examples are increasing the dimension of the z-layer to increase separability between mappings from different notes and/or using expert knowledge in post-processing the results of OCGP. We decided to leave such application-dependent detail out in this manuscript. Another issue that can be resolved at post-processing, is that the beginning of all notes are found with delay.

### V. EXPERIMENTING WITH TRUMPET NOTES

There are 35 trumpet notes taken from the University of Iowa Musical Instrument Samples (MIS): A3–A5, B3–B5, C4–C6, D4–D6, E3–E5, F3–F5, G3–G5, A3–A5s, C4s–C6s, D4s–D5s, F3s–F5s and G3s–G5s. The training clip consists of one instance of each of the E5, F5 and G5 notes with added noise. The test clip consists of all the trumpet notes in a random order. The system should be tested with different instances of the training notes, but the dataset consists of only one instance of each note. Therefore, for each E5, F5 and G5 notes, we created another 4 instances by adding noise to the original nearly noiseless instances from the dataset. We repeat the experiment for \(\sigma_{\text{noise}} = 0, \frac{1}{10}, \frac{1}{5} \ldots, \frac{1}{2}\) where \(\sigma_{\text{noise}}\) denotes the standard deviations of the added noise. The time domain sliding windows are preprocessed in the same manner as with the avian dataset. Also all the hyperparameters and post-processing are set as with the avian dataset. The resulting test confusion matrix with no added noise is illustrated in Table III. This perfect result on trumpet notes is due to their stationarity. For \(\sigma_{\text{noise}} = 0\), C6s gets mispredicted as E5. For \(\sigma_{\text{noise}} = \frac{1}{10}\) and \(\frac{1}{5}\), C6 note gets mispredicted as G5. At \(\sigma_{\text{noise}} = \frac{1}{2}\), the distinguishability between the 3 notes is mostly lost.

### VI. CONCLUSION

We described a system for simultaneous segmentation and classification of notes from recorded audio, in which HLDS was used to model the time-series representing notes. The

| babico-D | babico-F | pocaro-Z | outlier |
|----------|----------|----------|---------|
| 13/13    | 0        | 0        | 0       |
| babico-F | 0        | 4/4      | 0       | 0       |
| pocaro-Z | 0        | 0        | 2/2     | 0       |
| babico-A | 0        | 0        | 3/4     | 4/4     |
| babico-Z | 0        | 0        | 0       | 1/4     |
| pocaro-C | 0        | 0        | 0       | 6/6     |
| pocaro-D | 1/7      | 4/7      | 0       | 2/7     |
| pocaro-E | 0        | 0        | 0       | 10/10   |

**TABLE I**

Test confusion matrix for experimenting with avian notes. The row titles are the true labels and the column titles are the predictions.

| E5   | F5   | G5   | outlier |
|------|------|------|---------|
| 0    | 4/4  | 0    | 0       |
| 0    | 0    | 4/4  | 0       |
| 0    | 0    | 4/4  | 0       |
| 0    | 0    | 0    | 3/2/2   |

**TABLE II**

Test confusion matrix for experimenting with trumpet notes from MIS dataset. Outliers are all 32 notes other than E5, F5 and G5.
results of finding avian notes from recorded audio is promising. We proposed a novel method for parameter setting of HLDS. In future, we will compare HLDS with constrained non-negative matrix factorization which achieves spectral decomposition for multiple pitch estimation with state-space larger than the output space [5].

VII. FUTURE RESEARCH

Here is an outline of possible directions in which future research can be conducted:

- Incorporating nonlinearity to HLDS in a principled fashion. Some efforts have been conducted by incorporating correntropy [6] to HLDS [7].
- Automation of hyperparameter setting
- Learning innovation and error parameters according to data (Section III-B) and
- Learning coupling matrices, instead of setting them to fixed values, but with more constraints to avoid overfitting

REFERENCES

[1] G. T. Cinar, P. M. Sequeira, and J. C. Principe, “Hierarchical linear dynamical systems for unsupervised musical note recognition,” Journal of the Franklin Institute, vol. 355, no. 4, pp. 1638–1662, 2018.
[2] E. Coviello, A. B. Chan, and G. Lanckriet, “Time series models for semantic music annotation,” IEEE Transactions on Audio, Speech, and Language Processing, vol. 19, no. 5, pp. 1343–1359, 2010.
[3] A. B. Chan and N. Vasconcelos, “Layered dynamic textures,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 10, pp. 1862–1879, 2009.
[4] L. Kalantari, J. Principe, and K. Sieving, “Kernel mahalanobis distance with automated parameter setting for multiclass classification,” Submitted to IJCNN 2020.
[5] E. Vincent, N. Bertin, and R. Badeau, “Adaptive harmonic spectral decomposition for multiple pitch estimation,” IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 3, pp. 528–537, 2009.
[6] W. Liu, P. P. Pokharel, and J. C. Principe, “Correntropy: Properties and applications in non-gaussian signal processing,” IEEE Transactions on Signal Processing, vol. 55, no. 11, pp. 5286–5298, 2007.
[7] R. Singh and J. C. Principe, “Correntropy based hierarchical linear dynamical system for speech recognition,” in 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE, 2018.