Chord Function Recognition as Latent State Transition

Yui Uehara · Satoshi Tojo

Received: 22 July 2021 / Accepted: 29 August 2022 / Published online: 6 October 2022 © The Author(s) 2022

Abstract
This paper proposes an unsupervised learning of chord classification aiming at an autonomous recognition of chord functions. In this research, we employ hidden semi-Markov model to incorporate music metrical structure, and in addition, we combine the model with neural network components to embed context information such as beat positions and preceding chord sequences. Experimental results show that the added contexts considerably improve the perplexity. With the help of these neural networks, the proposed model automatically learns hidden states that appropriately represent chord categories. To this purpose, we pre-process the dataset minimally; that is, we only transpose pieces so as not to possess key signatures and ignore octave positions in pitch events. We observe the chord categories effectively cover chords that appeared in the corpus. We further show that the transitions between chord categories reflect the difference of tonalities with a tendency consistent with known chord functions.

Keywords Unsupervised learning · Hidden semi-Markov model · Neural network · Automatic chord segmentation · Chord function

Introduction

In music, combinations of tones form chords, and they progress in a somewhat regular way. These regularities have been studied as harmony theory [1–4], which is employed in artificial intelligence for music such as structural analysis [5, 6], recommendation [7], and music generation [8–10].

However, such textbook theories are too restrictive to represent diverse musical styles. Considering that these theories are derived from the experience of existing music, we are interested in whether a machine can induce inherent regularities in music by statistical learning. Unlike direct applications as music generation systems [11, 12], data-oriented, statistical acquisition of chord functions has been rarely investigated [13–17]. It has been reported that the statistically induced regularities considerably agree with known textbook theories, and it is expected to find fine-grained patterns that are characteristic of particular genres or composers [13–17].

These previous works, however, required the chord labeling or grouping as a pre-process, which includes chord segmentation [13, 16], modulation segmentation [14, 16], and chord annotation with scale degrees or Berklee chords [14, 15, 17]. Although harmony analyses in textbooks are based on given chord names, the correspondence between chord names and surface pitch events in scores is not trivial and diverse depending on musical styles. Thus, obtaining consistent Berklee annotations [18] or key assignments [19] were found to be more difficult than generally expected. In addition, pre-processed modulation detection or chord degree assignment ignores the mutual dependency of chord functions and keys. For example, G major chord is dominant in the C major key, while it becomes tonic in the G major key. Thus, the same chord may possess different regularities of progressions according to tonality [2]. Based on this knowledge, previous works for chord function identification have simplified the problem by removing the dependency for keys in advance. However, we regard that these functional changes of codes are also important features of music that should be captured in data-oriented statistical models.
We present a model that automatically learns chord categories and segments surface notes to be classified into the categories without relying on pre-defined chord symbols and key-assignments by utilizing an extension of hidden semi-Markov models (HSMM), where the chord categories are learned as hidden states that are used to predict the next coming chord in a chord sequence. As textbook theory required the concept of chord functions to explain chord progressions, we consider the characteristics of chords and the progressions between chords to be important musical features worth learning from the data.

Although our future goal is to apply the data-driven model to a variety of compositions to reveal the characteristics of individual composers, we start with J. S. Bach’s four-part chorales in this study. In this research, we avoid any heuristic pre-processing and only apply ones that are simply based on musical notation; in particular, we transpose pieces so as not to possess a key signature\(^1\) and ignore octave positions.

Technically, the task is a kind of sequential tagging, and thus it is close to the unsupervised part-of-speech (POS) tag induction in the natural language processing (NLP). We employ the unsupervised neural HMM, which is known to be one of the prominent models proposed for the POS tag induction [20]. The most advantageous feature of the neural HMM is that we can easily embed additional contexts. We customize the model and its additional contexts to be suitable for the music. Furthermore, to make the model be in accordance with metrical structures, we extend it to the neural hidden semi-Markov model (HSMM)\(^2\).

We use *perplexity* as the evaluation metric based on previous works [15, 17]. Experiments show that our model appropriately segments and classifies surface pitch-classes, especially with the smallest perplexity. Additional contexts with neural network modeling are shown to be effective to improve the perplexity. In addition, we show the transitions between categories reflect the difference of tonalities when we count them by separating pieces into groups of majors, minors, and *dorian* scales.

This paper is organized as follows. In Sect. “Related Work”, we review related studies. We introduce the proposed model in Sect. “Unsupervised Neural Hidden Semi-Markov Model for Chord Classification”. Then, we show the experimental results in Sect. “Experiments”. Finally, we summarize our contributions in Sect. “Conclusion”.

\(^1\) If the key signature is sharp, the overall pitch is lowered by 5 degrees (7 pitch-classes), and if it is flat, lowered by 4 degrees (5 pitch-classes). We repeat this as many times as the number of key signatures.

\(^2\) The code is available at https://github.com/yui-u/emerge-chord-function.

Related Work

Unsupervised learning of harmony or chord progressions has been studied through clustering methods [13, 14], HMMs [15–17], and Probabilistic Context-free Grammars (PCFG) [15]. These studies aimed at acquiring the progression rules, not based on possibly subjective human annotation, but relying on data-driven analysis.

They have found similarities between statistically induced clusters and chord functions in textbook theories, when compared models with the same number of clusters as the textbook [14–16]. Especially in HMM based models, not only obtained clusters but also their state transition property was found to have similarity to the known chord functions [16, 17]. However, selecting an optimal number of clusters would not be trivial. White and Quinn [16] proposed a methodology to find the best number of states for HMMs. They adopted the *k*-medoids clustering over hundreds of HMMs with different initial parameterizations. And then, they assumed that the number of hidden state $N$ that showed higher *silhouette width* (i.e., clearer cluster boundary) than $N−1$ and $N+1$ was appropriate. Uehara et al. [17] proposed another tandem approach to find the number of hidden states and detect modulation segments. On the other hand, Jacoby et al. [14] investigated the balance of the model by introducing the optimal complexity-accuracy curve, rather than fixing a particular number of hidden states. Similarly, Tsushima et al. [15] evaluated the generative models by the perplexity, which is a common metric for estimating generalization performance of a statistical model, and found that the larger number of hidden states leads a better score. Based on the latter approaches, we train our model with multiple numbers of hidden states and evaluate them by perplexity.

Preceding studies mentioned focused on harmonic structure solely. In other words, the duration of chords was ignored. However, a recent work proposed a supervised learning of the combined model of harmony and rhythm, and reported the efficacy of rhythmical information [21]. We also incorporate metrical information into our unsupervised learning by a semi-Markov model. Not only employing a semi-Markov model, we adopt an extension with neural networks based on the neural HMM. An important strength of the neural HMM is the seamless integration with additional contexts. Tran et al. [20] introduced the two additional contexts: embedded feature of preceding observations by the Long-Short Term Memory (LSTM) [27] and morphological information via character Convolutional Neural Networks. Despite its simple framework, the neural HMM outperformed even the highly polished Bayesian mixture model [22], or the hierarchial Pitman-Yor Process HMM [23]. Additional contexts in the neural HMM are also can be used for the unsupervised neural HSMM in the same manner.
Unlike the Hybrid DNN(Deep Neural Network)-HMM [24] that converts a pre-trained GMM(Gaussian Mixture Model)-HMM to a DNN-HMM or the Tandem model that combines features by a supervised DNN [25] to an HMM, the neural HMM is seamless and fully unsupervised. Our extension, i.e., the unsupervised neural HSMM differs from the Recurrent HSMM that makes use of the bi-LSTM for the purpose of reducing the error in the variational approximation [26].

Unsupervised Neural Hidden Semi-Markov Model for Chord Classification

Hidden Semi-Markov Model

We aim at presenting a model that predicts chord segments, chord categories and chord progressions simultaneously. To achieve this, we employ hidden semi-Markov model (HSMM), which is an extension of hidden Markov model (HMM), where lies a Markov chain of hidden states behind an observable sequence. The notion of the duration of a hidden state is introduced by the “semi-Markov” extension [28, 31–33]. While there can be various ways to model the duration of a state, usually one of the following three models is used for computational efficiency [34]: Explicit duration HMM [28, 31], Variable transition HMM [32], and Residential-time HMM [33]. We select the Residential-time HMM [33, 34] since the computational complexity of which is the smallest of the three.

Residential-time HMM assumes that a hidden state transition is independent of the duration of the previous hidden state and the duration of a hidden state is a discrete random variable. With this assumption, hidden state transition is described as follows

\[ P(Q_t = (j, \tau')|Q_{t-1} = (i, \tau)) = \begin{cases} a_{ij,\tau'} & \text{if } \tau = 1 \text{(transition)} \\ 1(\tau' = \tau - 1) & \text{if } \tau > 1 \text{(decrement)} \end{cases} \]

where \( i, j \in \{0, 1, ..., S\} \) are state indices, \( S \) is the number of hidden states, \( \tau \in \{1, 2, ..., D\} \) is the discrete duration time, and \( D \) is the maximum duration of a hidden state. We call a particular value of \( i \) or \( j \) as a hidden state index. While a hidden state index itself is just a serial number in hidden states of H(S)MM, a particular hidden state is expected to be a chord category after the training. Furthermore, \( a_{ij,\tau'} \) can be decomposed into transition probability and duration probability as follows.

\[ a_{ij,\tau'} = a_{ij}p_{j,\tau'} \]

Note that a hidden state changes to another only when the remaining duration \( \tau = 1 \). Therefore, the duration probability determines a hidden state duration, and the self-transition probability \( a_{ii} \) is always zero.

As is shown in the graphical representation of HSMM (Fig. 1a), each hidden state \((z_t = i)\) can produce multiple observations. Then, when a hidden state changes to another one \((j)\), the model calculates the duration probability of the next state \( p_{j,\tau'} \) as well as the transition probability \( a_{ij} \). However, there could be multiple possible combinations of the hidden state and the duration. In Fig. 1b, the number of hidden states is 3 and the maximum duration of a hidden state is 2; when the hidden state index is \( i = 0 \) and (remaining) duration is \( \tau = 1 \) at the time step \( t \), possible conditions for the previous time step\( t - 1 \) are: \((i = 0, \tau = 2)\), \((i = 1, \tau = 1)\), and \((i = 2, \tau = 1)\). Note that if the current remaining duration is \( \tau = 1 \), the hidden state must change to another one (dashed lines) at the next time step. On the other hand, if \( \tau > 1 \), the hidden state continues into the next time step (solid lines).

Framework

We expect that hidden states represent chord categories, which are not given a priori and thus learned in an unsupervised manner. With the Markov property assumption, hidden states (chord categories) are learned to govern the progression of the forthcoming chord category. We argue that an appropriate set of chord symbols would be diverse along with targeted pieces and thus adopt unsupervised learning.

We give a more detailed description of the proposed framework by showing an example in Fig. 2. Since HMM (HSMM) takes discrete time series, we set a time step by every sixteenth note, which is the minimum duration of note in the corpus, except for quite few exceptions of 32nd notes. We obtain a pitch-class vector for each segment. For example, when the time step is at 25, the pitch names of notes contained in the segment are \((C, D, G)\), and thus corresponding pitch class vector is \((0, 2, 7)\).

We build a table of vocabulary of observations, and call them \( i.e. \), tokens, from every combination of the four-part pitch-classes. For example, the token index for \((0, 2, 7)\) is set to 16. The whole vocabulary is shown in Appendix

\[ 0, 1, 2, ..., 11 \]

\[ 0, 1, 2, ..., 11 \]
As shown in the example, observations may include passing tones. We expect that a hidden state works as a chord category behind these raw pitch-class vectors. Looking at time step 25 again, we can see that the index of the hidden state is 7. Here, the hidden state is lasting from time step 25 to 28, which includes three types of pitch class vectors, (0, 2, 7), (2, 7, 11) and (2, 5, 7, 11), but all the three can be interpreted as a part of G major chord.

A hidden state emits a token with probability \( b_{ik} \), where \( i \) is a hidden state index and \( k \) is a token index. In the case of time step 25, pitch-class vector (0, 2, 7) (the token index of which is 16) is emitted with the probability of \( b_{7,16} \), from the hidden state indexed 7. A hidden state is changed to another one with the probability of transition probability \( a_{ij} \) multiplied by duration probability \( p_{jr} \), where \( i \) is the index of preceding hidden state, \( j \) and \( r \) are the index and duration of the forthcoming hidden state respectively. The probability of the transition from hidden state 7 to hidden state 3 at time step 28 to 29 is then \( a_{7,3}p_{3,8} \).

Note again that hidden states and their duration are not given \emph{a priori}, but obtained by unsupervised learning. More precisely, we first optimize parameters that determine transition, duration, and emission categorical distributions so that a tuned model gives higher likelihoods to targeted pieces, and then obtain the best hidden state sequence based on the model. Next, we give a detailed description of the architecture of neural networks used in the neural HSMM in the following section.

**Architecture of Neural Hidden Semi-Markov Model**

Different from the conventional HSMM that has categorical parameters just as matrices of parameters, \emph{neural} HSMM equips neural network components as functions for calculating transition, duration, and emission categorical distributions. As mentioned in the previous section, the role of each distribution is as follows.

---

Fig. 1 Hidden Semi-Markov model

(a) Graphical representation of Hidden Semi-Markov Model (Residential-time HMM). The notation \( z_{[t]} \) means a hidden state \( z \) ends at time \( t \), similarly, \( z_{[t+1:t+\tau]} \) starts at time \( t+1 \) and ends at time \( t+\tau \).

(b) Possible paths of hidden states and durations for HSMM (Residential-time HMM). The number of hidden states is 3 and the maximum duration of a hidden state is 2. The dashed line represents the state index change and the duration selection. The solid line represents the decrement of the state duration.
Fig. 2 BWV294 (in evaluation set) of analysis by the proposed model (8-state HSMM). The key is transposed to have no key signature. \(i, j\) are hidden states, which are uniquely determined by selecting the sequence with the maximum likelihood by the Viterbi algorithm after training. A token index \(k\) corresponds to a particular pitch observation. We show the lookup table of token indices in Table 4 in Appendix. Residential time \(r\) represents a remaining duration of a hidden state \(i\), which decreases by 1 in the same hidden state. Time steps increment by the sixteenth-note width segment. \nThe notation \(z_{13:16}=2\) represents that the hidden state index is \(i=2\) on time steps 13-16. \(a_{ij}\) is a transition probability and \(a_{2,3}\) represent that of hidden state index 2 to 3. \(b_{k,i}\) is an emission probability and \(b_{2,12}\) is that of token index 12 emitted from hidden state index 2. \(p_{3}\) is a duration probability and \(p_{1,3}\) represents that the next hidden state index is 3 and the duration of which is 2.

Hidden State Transition Probability

We denote a hidden state at time step \(t\) as \(z_t\) and state index of which as \(i\) or \(j\). Given the number of hidden state \(S\), each hidden state \(i \in S\) has a transition distribution to the next hidden state \(j \in S\), which is denoted as \(a_{ij}\), and thus \(\sum_j a_{ij} = 1\). Note that, in HSMM (Residential-time HMM), the self-transition \(a_{ii}\) is set to zero, since not transition distribution but duration distribution manages the duration of a hidden state. Instead of having \(a_{ij}\) just as a single learnable value, we prepare a function that calculates it by employing neural network components as follows.

\[
a_{ij} = P(z_{t+1} = j|z_t = i) = \text{softmax}_j(\text{MLP}_3([s;c_i]))
\]

where MLP_3 is a 3-layer Multi Layer Perceptron that has one input layer, two hidden layers and one output layer with a hyperbolic tangent (tanh) activation function after each hidden layer. The output layer of this function is softmax, to satisfy the condition \(\sum a_{ij} = 1\). The output layer size of the transition MLP_3 in (1) should be one smaller than the number of states \((S - 1)\), since it does not allow self-transitions.

\[\text{softmax}_j(x) = \frac{\exp(x_j)}{\sum \exp(x_i)}, \quad x = (x_0, \ldots, x_K).\]

Summary of notations (Table 5) and model settings (Table 6) can be found in Appendix.
Since $a_{ij}$ corresponds to the transition probability of hidden state $i$ to $j$, the input for the network includes a feature for hidden state $i$. We provide a hidden state embedding $s_i$, which is a learnable vector associated to hidden state index $i$. This hidden state embedding is also jointly used in the network for duration and emission probability (described in Sects. “Duration Probability” and “Emission Probability”), and would help to capture relationships between hidden states (chord categories) and emissions (observed pitch-class vectors).

Taking further advantage of neural HMM, we introduce two types of additional contexts for the calculation of hidden state transition probability, i.e., a pitch-class histogram and an embedded feature of preceding pitch-class vectors, and feed a concatenation of the two contexts $c_t$ to (1).

Additional context 1: Pitch-class histogram (HISTO) The first additional context is a pitch-class histogram that is the frequency of occurrence of each pitch-classes calculated from an entire phrase, which represents tonality. We split a piece into chord sequences at each fermata (point d’orgue) that works as a full-stop marker of a lyric, and then sum up the duration of each pitch-class to obtain a pitch-class histogram of a sequence. Even though the main tonality of a piece would be easily distinguished by the key signature, there would be local modulations and thus we employ such a histogram as an indirect information of local tonality. We obtain the feature of a pitch-class histogram of a sequence $r_{\text{hista}}$ as follows.

$$r_{\text{hista}} = \text{MLP}_2(v_{\text{hista}})$$

(2)

where $v_{\text{hista}} \in \mathbb{R}^{12}$ is the raw pitch-class histogram of a sequence.

10 However, the main tonalities of pieces that are written in church modes (e.g., dorian) would not correspond to the key signature system of modern tonality. In addition, major/minor is not distinguished by the key signature solely. Furthermore, some minor pieces retain the feature of church modes [3] and frequently proceed relative major keys [13, 17].

Additional context 2: Embedded feature of preceding observations by the LSTM (LSTM) The second additional context is an embedded feature of preceding observations by the Long-Short Term Memory (LSTM) [27], based on the idea from [20]. LSTM is a variant of Recurrent Neural Network (RNN) that recursively feeds observations, and thus it is suitable for a feature function for sequential data. In our case, it corresponds to a feature of a temporal sequence (from $t = 1$ to current time step $i$) of pitch-class vectors. At each time step, we input an observation embedding $o_k$ that associated with the observation $v_{\text{pitch}}^k$ at $t^{11}$ to LSTM as follows (3)(4).

$$o_k = \tanh(\text{MLP}_2(v_{\text{pitch}}^k))$$

(3)

$$h_t = \text{LSTM}(o_k, h_{t-1})$$

(4)

In calculation of an observation embedding (3), $v_{\text{pitch}}^k$ is a binary pitch-class vector, that is a 12-dimensional vector consisting of 1/0. For example, if the pitch-class vector is (2, 5, 7, 11), the corresponding binary pitch-class vector is (0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1). We expect that binary pitch-class vectors give raw information of observations, instead of symbolized token indices, and it is also used in emission distribution network described in Sect. “Emission Probability”.

Finally, we concatenate the two contexts and obtain the context vector as (5), where $[;]$ denotes vector concatenation. This context vector is applied to the transition probability network (1).

$$c_t = [r_{\text{hista}}, h_t]$$

(5)

We illustrate the network for calculating transition probability $a_{ij}$ in Fig. 3, which we have described in this section. Note that transition probability thus dynamically varies by
contexts in neural HSMM, though it is static in conventional models. Therefore, while we train a model without distinguishing major/minor pieces or local modulations, we expect that the additional contexts help to automatically adjust transition probabilities.

Initial Hidden State Probability

Since the first pitch-class vector at time step 1 does not have a preceding observation, this special distribution for the initial hidden state is provided. We set an external initial hidden state \( z_0 \) at time step 0 that has no emission and duration 1, to meet the initial boundary condition of HSMM. The initial hidden state probability \( \rho_i \) is calculated as follows. This is the only component remained almost the same as the conventional model except that it is normalized by the softmax function.

\[
\rho_i = P(z_0 = i) = \text{softmax}(\mathbf{x})
\]

where \( i \) is a hidden state index, \( \mathbf{x} \in \mathbb{R}^S \) is the learnable weight vector, and \( S \) is the number of hidden states.

Duration Probability

The duration probability determines how long the hidden state \( i \) (i.e., a chord category) will reside in the same hidden state. The network for calculating it is as follows.

\[
\rho_t = \text{softmax}(\text{MLP}_3([\mathbf{s}_t, \mathbf{v}_t^{\text{beat}}]))
\]

(6)

\[
\mathbf{r}_t^{\text{beat}} = \text{MLP}_2([v_{\text{timesig}}, \mathbf{v}_t^{\text{beat}}])
\]

(7)

Additional context 3: Beat position (BEAT) In (6), \( \mathbf{r}_t^{\text{beat}} \) is an additional context for the calculation of duration probability, which is expected to give a metrical information. \( [v_{\text{timesig}}, \mathbf{v}_t^{\text{beat}}] \) is a 2-dimensional vector that consists of a beat position and a numerator of the time signature. In this study, \( v_{\text{timesig}} \) is constant (= 4) since we only treat four-four time (4/4) pieces, while \( \mathbf{v}_t^{\text{beat}} \) could vary among \{1.0, 1.25, ..., 4.5, 4.75\}.

The output layer size of the duration MLP\(_2\) in (6) is 16 that is a maximum duration length that corresponds to a whole note. The hidden state embedding \( \mathbf{s}_t \) that is used in the network for transition probability is used here again, and is expected to help to jointly learn the transition and duration distribution of hidden state \( i \).

Emission Probability

We employ a discrete HSMM in this study, therefore, each observation is associated with token index \( k \in V \) in the vocabulary as mentioned in the example in Sect. “Framework”. The probability that hidden state \( i \) yields an token index \( k \) (\( b_{ik} \)) is calculated as follows, based on [20].

\[
b_{ik} = P(x_t = k | z_t = i) = \text{softmax}(\mathbf{s}_t^\top \mathbf{a}_k + l_k)
\]

\[
= \frac{\exp (\mathbf{s}_t^\top \mathbf{a}_k + l_k)}{\sum_{k'} \exp (\mathbf{s}_t^\top \mathbf{a}_{k'} + l_k)}
\]

(8)

Additional context 4: Observation embedding from binary pitch-class vector (PITCH) We use the same observation embedding \( \mathbf{a}_k \) as one used in the context for transition probability (3) for the calculation of emission probability (8). Instead of employing weight matrices or MLPs, the emission probability is learned as the dot product of a hidden state embedding and an observation embedding. The bias value \( l_k \in \mathbb{R} \) would help to consider the frequency of each chord. In contrast to the conventional discrete HMM that cannot use the raw information about the observation once converted into a token index, our model does not lose it by the observation embedding.

Training

Optimization

It is known that there is no analytical solution to obtain an optimal parameterization of an HMM that maximizes the likelihood of the observed sequence [28]. Although the widely used Baum-Welch re-estimation algorithm (that is a kind of the expectation-maximization algorithm) is able to obtain a locally optimal parameterization, it would be stuck in bad local optima when the likelihood surface is complex [28]. On the other hand, we can utilize gradient based methods to maximize the likelihood or the marginal probability \( \sum_{\mathbf{S}} \ln P(\mathbf{x}_{1:T}) \) where \( \mathbf{x}_{1:T} \) is an observed sequence since it is an optimization problem [28–30]. We employ the known dynamic programming named the forward algorithm for H(S)MMs to calculate the marginal probability [28, 34].

Neural networks models are generally trained by gradient based optimizers as well as the backpropagation for the gradient computing. Recently, along with the success of neural networks, efficient optimizers also have been proposed. By implementing the HSMM as a neural network, we can naturally utilize the gradient based optimization [29]. In this study, we use one of the latest optimizer RAma [35], which adjusts the learning rate automatically.

Forward Algorithm for Hidden Semi-Markov Models

To conduct the gradient based optimization, we calculate the marginal probability of an observed sequence \( \ln P(\mathbf{x}_{1:T}) \) by the forward algorithm, which marginalizes the possible
paths (as previously illustrated in Fig. 1b) of states and durations. We give the forward algorithm for the Residential-time HMM in the following.

To explain the duration of hidden states, we borrow the notation used by Yu et al. [34].

t_1 : t_2]: a hidden state lasts at latest from t_1 and ends at time t_2.

Among variants of HSMMs, the Residential-Time HMM does not allow the self-transition, and a transition to another hidden state is required when the “remaining” duration is \( \tau = 1 \). On the other hand, when \( \tau > 1 \), the duration just decrements at each time step.

The \( \alpha_i(j, \tau) \) represents the forward probability that the hidden state at \( t = j \) and the remaining duration of which is \( \tau \), and output tokens from \( t = 1 \) to \( t, i.e., x_{1:t} \) are observed. It is decomposed as follows.

\[
\alpha_i(j, \tau) = P(z_t=i) + j, x_{1:t})
\]

\[
= \alpha_{i-1}(j, \tau+1)P(x_t=k|z_t=j) + P(r|z_t=j)P(x_t=k|z_t=j)
\]

\[
\sum_{i,j} \alpha_{i-1}(i,1)P(z_t=j|z_{t-1}=i)
\]

\[
= \alpha_{i-1}(j, \tau+1)b_{jk} + p_{jk}b_{jk} \sum_{i,j} \alpha_{i-1}(i,1)a_{ij}
\]

where \( P(r|z_t=j) = p_{j\tau} \) is duration probability,\(^{12}\) \( P(x_t=k|z_t=j) = b_{jk} \) is emission probability, and \( P(z_t=j|z_{t-1}=i) = a_{ij} \) is transition probability. Note that when the state at \( t \) is \( j \), there are two possibilities: (i) the hidden state at time \( t-1 \) is also \( j \) with the remaining duration at there is \( \tau + 1 \), (ii) the hidden state at time \( t-1 \) is \( i \) (\( i \neq j \)) and the remaining duration is 1 then is transferred another hidden state \( j \) at the time \( t \).

We apply scaling by replacing \( \alpha_i(i, \tau) \) to the conditional probability, similar to HMM [34], and obtain a modified forward algorithm.

\[
\hat{\alpha}_i(j, \tau) = P(z_{t+\tau+1}=j|x_{1:t}) = \frac{\alpha_i(j, \tau)}{P(x_1, \ldots, x_t)}
\]

\[
C_i = P(x_1, \ldots, x_{t-1})
\]

\[
P(x_1, \ldots, x_t) = \prod_{t} C_t
\]

\[
C_t \hat{\alpha}_i(j, \tau) = \frac{\alpha_i(j, \tau)}{P(x_1, \ldots, x_{t-1})}
\]

\[
= \hat{\alpha}_{i-1}(j, \tau+1)b_{jk} + p_{jk}b_{jk} \sum_{i,j} \hat{\alpha}_{i-1}(i,1)a_{ij}
\]

As can be seen from (9), the marginal probability is obtained by \( P(x_1, \ldots, x_T) = \prod_{t} C_t \), where \( T \) is the length of an entire observation sequence, which means we can train the model only by executing the forward algorithm, without the backward algorithm usually used in EM algorithm. Note that \( C_t \) is obtained by summing up (10) about all \( j \) and \( \tau \), since \( \sum \sum \hat{\alpha}_i(j, \tau) = 1 \). Unlike HMMs, we set the initial probability to yield the initial hidden state \( z_0 \) that does not yield observation by considering the initial boundary condition: \( a_0(i, 1) = P(x_0 = i) \) and \( a_0(i, \tau) = 0 \) for \( \tau > 1 \) [34]. Therefore, \( a_0 = \hat{a}_0 \) and \( C_0 = 1.0 \) at \( t = 0 \).

### Experiments

In this section, we show the experimental results. We first describe the dataset (Sect. “Dataset”) and experimental setups (Sect. “Experimental Setups”). Thereafter, we show the evaluation result by perplexity in Sect. “Evaluation by Perplexity”. We give qualitative analysis for obtained chord categories and their progressions in Sect. “Qualitative Analysis for Induced Clusters”. Finally, we show a detailed exemplification and discussion of BWV267 with reference of a human annotation in Sect. “Discussion on an Analysis by the Model”.

### Dataset

We use J.S. Bach’s four-part chorales by the Riemenschneider numbering system (1-371) from the Music21 Corpus [37] as our dataset resource. At first, we have removed duplicated pieces according to the analysis by Dahn [38]. We only

| Dataset | #pieces | (maj.) | (min.) | (dur.) | (other) | #phrases | #obs.(HSMM) |
|---------|---------|--------|--------|--------|---------|----------|-------------|
| Train   | 185     | (89)   | (70)   | (23)   | (3)     | 1155     | 38452       |
| Eval.   | 51      | (21)   | (23)   | (6)    | (1)     | 314      | 10408       |
| Test    | 54      | (28)   | (20)   | (5)    | (1)     | 350      | 11868       |

\(^{12} \tau \in D \) is a discrete value of remaining time steps, and \( D \) is the maximum duration.
contain four-four time (4/4) pieces\textsuperscript{13}, thus the number of pieces is 290 (Table 1). We regard each phrase as an independent sequence, however, when we provide these phrases to training, evaluation, and testing set, we give randomness over pieces (not over phrases), since one piece may contain similar phrases.

**Experimental Setups**

In this research, we disregard the difference of absolute pitch so that we transpose all the major pieces to C major and all the minor ones a minor. Some pieces of the corpus are written in dorian scale without key signature. From the viewpoint of modern tonality, we regard them in d minor; however, we do not shift them to a minor. Also, local modulations are shifted in a similar way, so as to preserve the relative positions of constituent notes. Finally, we ignore the difference by octave in pitch events and inversion in chords.

We assign token indices ($k$) for pitch-classes the accumulated pitch-duration of them in the dataset over 95%, and then, remained chords are merged to “Others”. The obtained vocabulary size was 80, including “Rest”.

In training, we set the mini-batch 8. We train models up to 500 epochs, however, the process is stopped when the lowest loss on the evaluation set is not updated over 20 epochs. We apply the dropout [36] to each hidden layer of MLPs, which randomly ignores some ratio of neurons (12.5% in our case) during training to avoid over-fitting.

**Evaluation by Perplexity**

**Evaluation Metric**

Following previous works [15, 17], we use perplexity as an evaluation metric, defined by the following equation.

$$P = \exp\left(\frac{-1}{T} \ln P(x_1, \ldots, x_T; \theta)\right)$$

(11)

where $(x_1, \ldots, x_T)$ is a sequence of output tokens and $\theta$ is a set of model parameters. The smaller perplexity for test data means the better generalization performance since perplexity corresponds to log-average of inverse probability, and thus it is commonly used for evaluating probabilistic models. We examine the performance on multiple numbers of hidden states: 3 to 16. We conduct the experiments with three different random seeds of {0, 1, 2} for each number of hidden states, and report the averaged scores in Sect. “Evaluation by Perplexity”.

\textsuperscript{13} We have also excluded 27 pieces that are not four-part voices or have some problems such as a collapsed format.
the perplexity. Among them, removing\(^{14}\) the observation representation by pitch-classes (–PITCH) led a significant drop. Since we do not employ MLPs for the architecture of calculation of emission probability but force a model to learn relationships between hidden states and vocabularies directly, the information of raw pitch-classes would help the learning.

\(^{14}\) In this case, we used an observation embedding of learnable weights in the same way of the hidden state embedding instead of the pitch-class one.

**Table 3** Chord categories obtained by the best scored 8-states neural HSMM. The chord category is named after the chord name of the top emission for each hidden state.

| Hidden state index | State0 | State1 | State2 | State3 |
|--------------------|--------|--------|--------|--------|
| Chord category     | Δ      | £      | £      | Δ      |
| emit. top3         | (1)    | (2)    | (3)    |        |
|                    | D.F.(Gb).A | C.(D#).E.A | D.F.A | C.E.G |
|                    | C.D.(G).A | C.♯(D#).E.G.A | C.D.F.A | C.E.G.B |
|                    | C.E.(G).A | C.(D#).E.G.A | C.D.F.A | C.E.G.B |

**Table 3** Chord categories obtained by the best scored 8-states base HSMM. The chord category is named after the chord name of the top emission for each hidden state.

| Hidden state index | State0 | State1 | State2 | State3 |
|--------------------|--------|--------|--------|--------|
| Chord category     | Δ      | £      | £      | Δ      |
| emit. top3         | (1)    | (2)    | (3)    |        |
|                    | C.F.A | E.G.(Ab).B | C.E.A | D.G.B |
|                    | Others | E.G.B | C.E.A | D.G.B |
|                    | C.E.F.A | D.E.G.(Ab).B | C.E.A.B | C.D.G |
In this section, we give qualitative analysis and discussion about the induced chord clusters. We focus on the eight-state models since we have transposed pieces not to have key signatures and have assumed that each cluster would roughly correspond to a triad on the diatonic scale, or otherwise to the rests.

**Induced Clusters and Model’s Perplexities**

We investigate if we can find clearer clusters in the higher scored model in obtained examples. We show the emission probabilities of the best neural HMM, the worst neural HSMM, and the best baseline model in Fig. 5. Note that we have executed three trials of training for each model with a different random seed among \{0, 1, 2\}. Here the “best” model means that the evaluation perplexity of which is the smallest among the three trials.

We observe that the clusters obtained by the best scored neural HMM mainly consist of the chords on diatonic scales (C major and a minor) and its commonly-used borrowed chords such as D major chord. We show the emission probabilities of the top three emissions for each hidden state index in Fig. 5 and summarize them in Table 3. For ease of readability, we name chord categories by the name of the most frequent output chord for each hidden state index. Although the most frequent chord is representative of the category, it does not correspond to a unique chord but a set of emission probabilities; therefore, we add “hat” to it. The cluster Â may emerge from two reasons: the common use of the Picardy third, or the dominant of dorian mode pieces. It is worth noting that seventh chords and passing chords are merged into appropriate clusters, e.g., C.D.E.G in the same hidden state for C.E.G (state3), and D.F.G.B for D.G.B (state7).

Even though employing the same neural HSMM, the worst perplexity model (in the middle of Fig. 5) seems to be less appropriate than the best model. For example, C.E.A is mixed up with C.F.A and E.G(A♭).B in state7, and C.E.G (state4) and passing chords around it (state0) are separated. Although we would be able to choose a model by perplexity, we admit that difficulty for obtaining the global optimum still exists even though we employ the efficient gradient based optimizer.

The best baseline model (the evaluation perplexity of which is 9.92) is still worse than the worst neural HSMM (9.18). Not only the best baseline model scored a worse perplexity, but it possessed more miscellaneous clusters than neural HSMMs. However, we can observe that C.E.G and D.F.G.B of state3, or D.G.B and D.F(G♭)A of state5, in the same tonality, would tend to be merged into the same category in the baseline HSMM.

---

**Qualitative Analysis for Induced Clusters**

In this section, we give qualitative analysis and discussion about the induced chord clusters. We focus on the eight-state models since we have transposed pieces not to have key signatures and have assumed that each cluster would roughly correspond to a triad on the diatonic scale, or otherwise to the rests.
Hidden State Transitions

Since chord functions lie in the regularity of chord progressions, we try to find them by investigating the transitions between obtained chord categories. We count the hidden state transitions on major pieces, minor pieces and dorian pieces separately to see whether the model appropriately reflects the difference of tonalities in its state transitions. Note that, instead of examining the hidden state transition probability directly, we count the number of transitions after decoding the sequence of hidden states by the Viterbi algorithm since the hidden state transition probability changes by contexts.

We show the result of hidden state transition properties by the best scored neural HSMM in Fig. 6 (the emission probability of which is shown at the top of Fig. 5). In major pieces, the tendency that \{state4:F \, \rightarrow \, state7:Ĝ\} (tonic) is noticeable. While in minor pieces, the tendency described above suggests existence of the relative major keys, which is consistent with previous works [13, 17]. In addition, a strong transition from state5:Ê (dominant) \rightarrow state6:â (tonic) are observed. Unlike in major pieces, the transition from state2:d (subdominant)
Fig. 8  Chord classification by the neural HSMM on BWV267 (phrase No.1–4)
Fig. 9  Chord classification by the neural HSMM on BWV267 (phrase No.5–8)
→ state5:Ê (dominant) is increased in minor pieces, since it corresponds to the transition of subdominant → dominant in a minor. Finally, in dorian pieces, transitions from state1:Â (dominant) → state2:d (tonic) are observed. Unlike C major and a minor pieces, the transition from state7:G no longer has the strong tendency of proceeding to state3:Ĉ but tends to proceed state1:Â instead, corresponding to the progression of subdominant → dominant.

Discussion on an Analysis by the Model

In this section, we discuss the adequacy of our model from an obtained result. We select BWV267 from our testing set, since a human analysis on which is publicly available [37]. Although our aim is not to reproduce the human analysis, we consult it as a possible interpretation for local modulations and chord annotations. Note that we normalized the score to have no key signature, and thus the main key became C major, while the original key was G major. We show extractions of the result in Fig. 7. According to the human analysis [37], there are local modulations to {F major, G major, d minor, g minor} keys in this piece.

In this example, we could see that the model found an effective set of clusters to cover chords appeared in a piece including local modulations. For example, in the section of G major key (phrase No.4, time step 21–32), we observed the cluster of D*(state0), which was a borrowed chord seen from C major key. This cluster D* was used again in the section of g minor key (phrase No.7, time step 7–32). Interestingly, g minor chords, i.e., D.G.A♭ (B♭) (k = 21) and D.F.G.A♭(B♭) (k = 60), were classified into the same cluster as G (state7), since g minor chord and G major one shared the dominant (state0: D). Similarly, in the section of F major (phrase No.4, time step 1–20), C major dominant seventh chords, i.e., C.E.G.A♭ (B♭) (k = 24), were classified into the same cluster as C major chords (state3: Ĉ). According to Schoenberg, local modulations basically remain in closely related keys and thus can be analyzed by altered chords [3]. In the case of g/G chords and C/C7 chords described above, these alternations (g or C7) do not change the probable progression of chords, and thus would be classified into the same function as the one on the diatonic scale (G or C).

In the longest modulation section of d minor (phrase No.6), we could see that the cluster of tonic chord (state3: Ĉ) has not appeared and that the cluster of d minor chord (state2:d) tended to proceed to A major chords (state1:Â).

Since we ignored inversions of chords in this study, we basically could not capture the difference between them. We admit that it needs further consideration, and left it as a future work. In addition, we observed more “Others” tokens (k = 79) in modulation sections that hindered the analysis. “Others” or “Unknown” token has been introduced in discrete HMMs to avoid useless increase of dimension of the emission distribution. However, once an observation is converted into “Others”, the raw information about it (which is pitch-class vector in our case) will be lost. More effective treatment for less frequent observations or more direct representation for emission distribution is another important future work for us.

Conclusion

In this paper, we aimed to obtain the regularity of chord progressions by the data-oriented unsupervised learning. Chord functions were introduced to explain typical progressions \textit{a posteriori}, and thus music is not constrained by such functions. Thus we have been interested in finding statistically plausible chord functions by machines.

Harmony analysis consists of the following four processes; (i) defining an appropriate set of chord labels (e.g., chord degrees or Berklee chords in traditional approaches), (ii) determining the scopes of chords and their labels from surface structures of a piece, (iii) identifying the sequence of chords based on the determined labels, (iv) and finally, analyzing the labeled chord sequence by examining transitions between chords, i.e., chord functions. Our objective is to let computers execute these processes, without relying on human annotations or pre-defined labels. In our experiments, we have only pre-processed the source scores by shifting keys either to C major or to a minor, and ignored the difference by octave in pitch events. The pre-process could be executed automatically without relying on external music feature extraction tools.

We regarded that the states in hidden Markov model (HMM) represented chord categories, and we employed an extended HMM with the notion of duration of a hidden state, called hidden semi-Markov model (HSMM). Furthermore, we utilized neural networks for calculating categorical distributions to embed the pitch-class distribution, preceding chord sequences, beat information, and so on, as additional contexts.

The experimental results satisfactorily showed the efficacy of such contexts in terms of perplexity in comparison with the baseline and other ablation studies (Sect. “Evaluation by Perplexity”). We successfully obtained clear chord clusters with the best scored proposed model where seventh chords and

---

15 \url{https://github.com/cuthbertLab/music21/tree/master/music21/corpus/bach/choraleAnalyses}
16 The result for the whole piece is shown in Appendix: Fig. 8 (phrase No.1–4) and Fig. 9 (phrase No.5–8).
borrowed chords were classified into appropriate clusters (Sect. “Induced Clusters and Model’s Perplexities”). Also, we have shown a full exemplification of BWV267 (Sect. “Discussion on an Analysis by the Model”).

As was mentioned by Hugo Riemann [2], “In the change of these function (tonic, dominant, and sub-dominant) lies the essence of modulation”. According to this, the distinction of keys should not be given prior to the chord analysis, but the modulations should be externalized as a result of the analysis. We followed this aphorism and have avoided the independent detection of modulation. Instead, we have characterized each hidden state, counting the number of their transitions, and dividing the experimental results into majors, minors, and dorian scale. In this regard, we have not automatically detected the keys, however, we have found that the distribution of state transitions were consistent with the known classical theory, in an unsupervised manner (Sect. “Hidden State Transitions”).

We plan to improve the proposed methodology to incorporate modulation detection, to integrate with the chord analysis, as our future work. Also, we intend to apply our model to pieces that may not be faithful to classical musicology. We need further extensions to overcome the out-of-vocabulary issue described in Sect. “Discussion on an Analysis by the Model” for pieces that accompany more elaborations than four-part chorales. In addition, there are usually few pieces in a particular genre by a specific composer. In such a case, learned parameters of transition probabilities and emission probabilities are expected to be good initial values (pre-trained information) for fine-tuning, which is known to be effective when a sufficient amount of training data is not available. For this reason, we used Bach’s four-part chorales that contain a substantial amount of coherently structured pieces composed at the beginning of the tonal music. We hope the future work of extensive music analysis by machines would contribute to expanding the scope of discovering cultural evolution in music.

A Appendices

See Tables 4, 5 and 6.
Table 5: Notations in the Neural HMM and HSMM

| N          | : number of sequences in a mini-batch |
|------------|--------------------------------------|
| n ∈ N      | : a sequence index in a mini-batch   |
| S          | : number of hidden states            |
| V          | : vocabulary size                    |
| T          | : maximum time step of a sequence    |
| t ∈ T      | : time step                          |
| z_t        | : hidden state at t                  |
| i, j ∈ S   | : state indices                      |
| D          | : maximum hidden state duration      |
| τ ∈ D      | : duration index                     |
| s_t        | : index of observed pitch-class content at t |
| k ∈ V      | : observed symbol index              |
| x_{1:T}    | : an observed sequence               |
| p_t        | : duration probability               |
| p_i        | : initial hidden state probability   |
| π          | : unnormalized initial hidden state probability |
| a_{ij}     | : hidden state transition probability |
| b_k        | : emission probability               |
| s_h        | : hidden state embedding             |
| v_{pitch}  | : binary pitch-class vector          |
| v_{histo}  | : observed pitch-class histogram     |
| r_{histo}  | : pitch-class histogram encoding     |
| o_k        | : observation encoding               |
| h_k        | : emission bias                      |

Table 6: The size of layers and related equations

| Layer Description | Size | Eq. |
|-------------------|------|-----|
| hidden layer size of MLP_{3} for transition and duration probability | 16 | (1), (6) |
| dimension of hidden state embedding (x_{i}) | 16 | (1), (6), (8) |
| hidden layer sizes of MLP_{2} for histogram embedding | 8 | (2) |
| hidden and layer sizes of MLP_{2} for observation embedding | 16 | (3) |
| dimension of observation embedding vector (o_{j}) | 16 | (3), (8) |
| dimension of LSTM hidden vector h_{j} | 16 | (4) |
| hidden layer size of MLP_{2} for beat encoder | 8 | (7) |
| dimension of beat embedding vector r_{beat}^{t} | 8 | (7) |

Funding This work is supported by JSPS Kaken Nos. 16H01744, 20H04302, and 21H03572.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Rameau JP. Treatise on Harmony. New York: Dover Publications; 1971.
2. Riemann H. Harmony Simplified: Or the Theory of the Tonal Functions of Chords. London: Augener; 1896.
3. Schoenberg A. Structural Functions of Harmony (revised edition). W. W. Norton & Company 1969
4. Mulholland J, Hojnacki T. The Berklee Book of Jazz Harmony, Berklee Press; 2013
5. Granroth–Wilding M, Steedman M. Statistical parsing for harmonic analysis of jazz chord sequences. In: International Computer Music Conference, 2012; pp.478–485
6. Rohrmeier M. Towards a generative syntax of tonal harmony. J Math Music. 2011;5(1):35–53.
7. Kuo F, Shan M. A personalized music filtering system based on melody style classification. In: Proceedings of the 2002 IEEE International Conference on Data Mining, 2002; pp.649–652
8. Groves R. Automatic Harmonization Using a Hidden Semi-Markov Model. In: AIIDE Workshop,2013; pp.48–54
9. Anders T, Eduardo RM. A computational model that generalises schoenberg’s guidelines for favourable chord progressions, In: 6th Sound and Music Computing Conference, 2009;pp.48–52
10. Navarro M, Cuatno M, Bernardes G, Nunez de Castro L, Corchado JM. In: International Conference on Evolutionary and Biologically Inspired Music and Art, 2015;pp.175–186
11. Hadjeres G, Pachet F, Nielsen F. Deepbach: a steerable model for bach chorales generation, In: International Conference on Machine Learning, 2017;pp.1362–1371
12. Huang C-Z A, Vaswani A, Uszkoreit J, Shazeer N, Hawthorne C, Dai A M, Hoffman M D, Eck D. Music Transformer: Generating Music with Long-Term Structure.,2018 arXiv preprint arXiv:1809.04281
13. Rohrmeier M, Cross I. Statistical properties of tonal harmony in Bach’s chorales, In: 10th International Conference on Music Perception and Cognition, 2008; pp. 619–627
14. Jacoby N, Tishby N, Tymoczko D. An information theoretic approach to chord categorization and functional harmony. J New Music Res. 2015;44(3):219–44.
15. Tsushima H, Nakamura E, Itoyama K, Yoshii K. Generative statistical models with self-emergent grammar of chord sequences. J New Music Res. 2018;47(3):226–48.
16. White CW, Quinn I. Chord context and harmonic function in tonal music. Music Theory Spectr. 2018;40(2):314–35.
17. Uehara Y, Nakamura E, Tojo S. Chord function identification with modulation detection based on HMM. In: Proceedings of 14th International Symposium on Computer Music Multidisciplinary Research, 2019; pp. 59–70.

18. Koops HV, de Haas WB, Burgoyne JA, Bransen J, Kent-Muller A, Volk A. Annotator subjectivity in harmony annotations of popular music. J New Music Res. 2019;48(3):232–52.

19. Temperley D. The tonal properties of pitch-class sets: Tonal implication, tonal ambiguity, and tonalness. Comput Musicol. 2007;15:24–38.

20. Tran K, Bisk Y, Vaswani A, Marcu D, Knight K. Unsupervised neural hidden markov models. In: Proceedings of the Workshop on Structured Prediction for NLP, 2016; pp. 63–71.

21. Harasim D, O’Donnell TJ, Rohrmeyer M. Harmonic syntax in time rhythm improves grammatical models of harm. In: Proceedings of the 20th International Conference on Music Information Retrieval, 2019; pp. 335–342.

22. Christodoulopoulos C, Goldwater S, Steedman M. A Bayesian mixture model for PoS induction using multiple features. In: Proceedings of the 2011 conference on empirical methods in Natural Language Processing, 2011; pp. 638–647.

23. Blunsom P, Cohn T. A hierarchical Pitman-Yor Process HMM for unsupervised part of speech induction. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011; pp. 865–874.

24. Yu D, Deng L, Dahl GE. Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition. In: NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning; 2010.

25. Hermansky H, Ellis DPW, Sharma S. Tandem connectionist feature extraction for conventional HMM systems. In: 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing, Proceedings (Cat. No.00CH37100), 2000; vol. 3, pp. 1635–1638.

26. Dai H, Dai B, Zhang YM, Li S, Song L. Recurrent hidden semi-markov model. In: Proceedings of the International Conference on Learning Representations; 2017.

27. Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Comput. 1997;9(8):1735–80.

28. Rabiner LR. A tutorial on hidden markov models and selected applications in speech recognition. Proc IEEE. 1989;77(2):257–86.

29. Kim Y, Wiseman S, Rush AM. A tutorial on deep latent variable models of natural language, 2018 arXiv preprint, arXiv:1812.06834.

30. Levinson SE, Rabiner LR, Sondhi MM. An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. Bell Syst Tech J. 1983;62(4):1035–74.

31. Ferguson JD. Variable duration models for speech, In Proc. Symposium on the Application of Hidden Markov Models to Text and Speech, 1980;143–179.

32. Levinson SE. Continuously variable duration hidden Markov models for automatic speech recognition. Comput Speech Lang. 1986;1(1):29–45.

33. Yu SZ, Kobayashi H. An efficient forward-backward algorithm for an explicit-duration hidden Markov model. IEEE Signal Process Lett. 2003;10(1):11–4.

34. Yu SZ. Hidden semi-Markov models. Artif Intell. 2010;174(2):215–43.

35. Liu L, Jiang H, He P, Chen W, Liu X, Gao J, Han J. On the variance of the adaptive learning rate and beyond. In: International Conference on Learning Representations; 2020.

36. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors, CoRR abs/1207.0580; 2012.

37. Cuthbert MS, Ariza C. music21: a toolkit for computer-aided musicology and symbolic music data. In: Proceedings of the 11th International Society for Music Information Retrieval Conference; 2010.

38. Dahn L. So how many Bach four-part chorales are there?, http://www.bach-chorales.com/HowManyChorales.htm.; 2018

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.