MODELING SPEECH PARAMETER SEQUENCES WITH LATENT TRAJECTORY HIDDEN MARKOV MODEL

Hirokazu Kameoka

Nippon Telegraph and Telephone Corporation / The University of Tokyo

ABSTRACT

This paper proposes a probabilistic generative model of a sequence of vectors called the latent trajectory hidden Markov model (HMM). While a conventional HMM is only capable of describing piecewise stationary sequences of data vectors, the proposed model is capable of describing continuously time-varying sequences of data vectors, governed by discrete hidden states. This feature is noteworthy in that it can be used to model many kinds of time series data that are continuous in nature such as speech spectra. Given a sequence of observed data, the optimal state sequence can be decoded using the expectation-maximization (EM) algorithm. Given a set of training examples, the underlying model parameters can be trained by either the expectation-maximization algorithm or the variational inference algorithm.

Index Terms— Sequential modeling, Hidden Markov model (HMM), Trajectory HMM, Latent trajectory HMM, Expectation-Maximization algorithm, variational inference

1. INTRODUCTION

The weakness of hidden Markov models (HMMs) is that they have difficulty in modeling and capturing the local dynamics of feature sequences due to the piecewise stationarity assumption and the conditional independence assumption on feature sequences. Traditionally, in speech recognition systems, this limitation has been circumvented by appending dynamic (delta and delta-delta) components to the feature vectors. HMM-based speech synthesis systems [1] also use the joint vector of static and dynamic features as an observed vector in the training process. In the synthesis process, on the other hand, a sequence of static features is generated according to the output probabilities of the trained HMM given an input sentence by taking account of the explicit constraint between the static and dynamic features [2]. Although the HMM-based speech synthesis framework has many attractive features, one drawback is that the criteria used for training and synthesis are inconsistent. While the joint likelihood of static and dynamic features is maximized during the training process, the likelihood of only the static features is maximized during the synthesis process. This implies that the model parameters are not trained in such a way that the generated parameter sequences become optimal. To address this problem, Zen [3] introduced a variant of HMM called the "trajectory HMM," which was obtained by incorporating the explicit relationship between static and dynamic features into the traditional HMM. This has made it possible to provide a unified framework for the training and synthesis of speech parameter sequences, however, it causes difficulty as regards parameter inference. Since the conditional independence assumption on the feature vectors is lost, efficient algorithms for training and decoding regular HMMs such as the Viterbi algorithm and the Forward-Backward algorithm are no longer applicable to the trajectory HMM. Thus, some approximations and brute-force methods are usually necessary to obtain training and decoding algorithms [3, 4].

In this paper, we propose formulating a new model called the "latent trajectory HMM." In contrast with the conventional trajectory HMM, the present model splits the generative process of an observed feature sequence into two processes, one for a sequence of the joint vectors of static and dynamic features given HMM states and the other for an observed feature sequence given the sequence of the joint vectors. By treating the joint vector of static and dynamic features as a latent variable to be marginalized out, we obtain a probability density function of an observed feature sequence with a different form from the likelihood function of the trajectory HMM. As described below, this new formulation naturally allows the combined use of powerful inference techniques such as the expectation-maximization (EM) algorithm, Viterbi algorithm and Forward-Backward algorithm for training and decoding, while still retaining the spirit of the original trajectory HMM.

This work is not only directed towards speech synthesis applications but also towards several different applications such as voice conversion and acoustic-to-articulatory mapping, in which trajectory modeling has proven to be effective [5–7]. Another interesting application we have in mind is audio source separation. Recently, we proposed methods for single- and multi-channel audio source separation based on factorial HMMs [8–11], where the spectrogram of a mixture signal is modeled as the sum of the outputs emitted from multiple HMMs, each representing the spectrogram of an underlying source. One promising way to improve this approach would be to incorporate the dynamics of source spectra. This can be accomplished by plugging the present model into the factorial HMM formulation. The present formulation will play a key role in making this possible.

This work was supported by JSPS KAKENHI Grant Numbers 26730100 and 26280060. The author would like to thank Dr. Tomoki Toda (NAIST) for fruitful discussions and Mr. Tomohiko Nakamura (University of Tokyo) for his help with conducting the experiments.

978-1-4673-7454-5/15/$31.00 ©2015 IEEE

2. TRAJECTORY HIDDEN MARKOV MODEL

We start by briefly reviewing the original formulation of the trajectory HMM [3]. Let us use $\boldsymbol{c}_t$ to denote a D-dimensional static feature vector and define the joint vector of $\boldsymbol{c}_t$ and its velocity and acceleration components $\boldsymbol{o}_t := [\boldsymbol{c}_t^\mathsf{T}, \Delta\boldsymbol{c}_t^\mathsf{T}, \Delta^2\boldsymbol{c}_t^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{3D}$ as the observed vector at time t. We write the sequences of the static features and the observed vectors as $\boldsymbol{c} = [\boldsymbol{c}_1^\mathsf{T}, \ldots, \boldsymbol{c}_T^\mathsf{T}]^\mathsf{T}$ and $\boldsymbol{o} = [\boldsymbol{o}_1^\mathsf{T}, \ldots, \boldsymbol{o}_T^\mathsf{T}]^\mathsf{T}$, respectively. Thus, the dimensions of $\boldsymbol{c}$ and $\boldsymbol{o}$ become DT and $3DT$. The relationship between $\boldsymbol{c}$ and $\boldsymbol{o}$ can be described explicitly using a constant $3DT$ by DT matrix $\boldsymbol{W}$ as

$$\boldsymbol{o} = \boldsymbol{W}\boldsymbol{c}, \tag{1}$$

where $\boldsymbol{W}$ is a sparse matrix that appends first and second order time derivatives to the static feature vector sequence.

Within the traditional HMM framework, a sequence of observed vectors, $\boldsymbol{o}$, is simply assumed to be generated from an HMM. Here, if we assume the emission probability density to be a single Gaussian distribution, the probability density function of $\boldsymbol{o}$ given a state sequence $\boldsymbol{s} = [s_1, \ldots, s_T]$ and an HMM parameter set $\boldsymbol{\lambda} = \{\boldsymbol{\mu}, \boldsymbol{U}, \boldsymbol{\pi}\}$, with $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}_{1\le i\le I}$, $\boldsymbol{U} = \{\boldsymbol{U}_i\}_{1\le i\le I}$, and $\boldsymbol{\pi} = \{\boldsymbol{\pi}_{i,j}\}_{1\le i\le I, 1\le j\le J}$, is given as

$$p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{o};\boldsymbol{\mu}_{\boldsymbol{s}},\boldsymbol{U}_{\boldsymbol{s}}) = \prod_{t=1}^{T}\mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_{s_t},\boldsymbol{U}_{s_t}), \tag{2}$$

where $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})$ denotes a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$:

$$\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) \propto \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}. \tag{3}$$

$\boldsymbol{\mu}_{\boldsymbol{s}}$ and $\boldsymbol{U}_{\boldsymbol{s}}$ denote the mean sequence and a block diagonal matrix whose diagonal elements are given by the sequence of the covariance matrices of the emission densities over time:

$$\boldsymbol{\mu}_{\boldsymbol{s}} = [\boldsymbol{\mu}_{s_1}^\mathsf{T}, \ldots, \boldsymbol{\mu}_{s_T}^\mathsf{T}]^\mathsf{T}, \tag{4}$$
$$\boldsymbol{U}_{\boldsymbol{s}} = \mathrm{diag}(\boldsymbol{U}_{s_1}, \ldots, \boldsymbol{U}_{s_T}). \tag{5}$$

In HMM-based speech synthesis systems, the parameter set is typically trained by solving the maximum likelihood estimation problem

$$\hat{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\mathrm{argmax}}\ \log\sum_{\boldsymbol{s}} p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})p(\boldsymbol{s}), \tag{6}$$

where $p(\boldsymbol{s})$ is given by the product of the state transition probabilities. At the synthesis stage, given a state sequence $\boldsymbol{s}$ and with the trained parameter $\boldsymbol{\lambda}$, a static feature sequence $\boldsymbol{c}$ is generated according to

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}}{\mathrm{argmax}}\, p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda}), \tag{7}$$

where $p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda})$ is defined as

$$\begin{aligned} p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda}) &\propto \mathcal{N}(\boldsymbol{W}\boldsymbol{c};\boldsymbol{\mu}_{\boldsymbol{s}},\boldsymbol{U}_{\boldsymbol{s}}) && (8)\\ &\propto e^{-\frac{1}{2}(\boldsymbol{c}^\mathsf{T}\boldsymbol{W}^\mathsf{T}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{W}\boldsymbol{c} - 2\boldsymbol{c}^\mathsf{T}\boldsymbol{W}^\mathsf{T}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{\mu}_{\boldsymbol{s}})} && (9)\\ &= \mathcal{N}(\boldsymbol{c};\bar{\boldsymbol{c}}_{\boldsymbol{s}},\boldsymbol{V}_{\boldsymbol{s}}). && (10) \end{aligned}$$

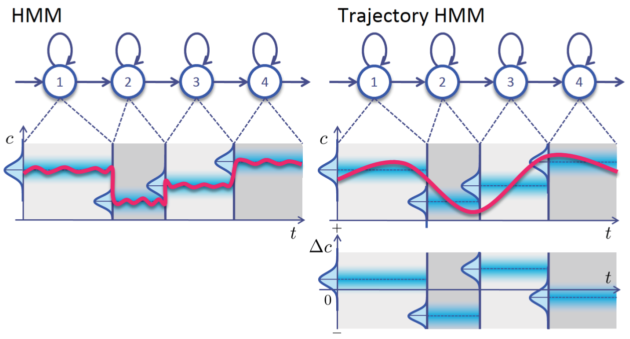

Fig. 1. Illustrations that show the difference between HMM and trajectory HMM.

By completing the square in the exponent of (9) with respect to $\boldsymbol{c}$, we immediately obtain $\bar{\boldsymbol{c}}_{\boldsymbol{s}}$ and $\boldsymbol{V}_{\boldsymbol{s}}$ as

$$\bar{\boldsymbol{c}}_{\boldsymbol{s}} = (\boldsymbol{W}^\mathsf{T}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{W})^{-1}\boldsymbol{W}^\mathsf{T}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{\mu}_{\boldsymbol{s}}, \tag{11}$$
$$\boldsymbol{V}_{\boldsymbol{s}} = (\boldsymbol{W}^\mathsf{T}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{W})^{-1}. \tag{12}$$

Thus, the solution to (7) is $\bar{\boldsymbol{c}}_{\boldsymbol{s}}$. Geometrically, (10) can be viewed as a cutting plane of the density $p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})$ at $\boldsymbol{o} = \boldsymbol{W}\boldsymbol{c}$.

As shown above, the traditional HMM-based framework uses different criteria for training and synthesis: While $p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})$ is used for training, $p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda})$ is used for synthesis. This implies that $\hat{\boldsymbol{\lambda}}$ is not necessarily optimal for generating optimal $\boldsymbol{c}$. To address this inconsistency between the training and synthesis criteria, Zen [3] proposed introducing a framework called the "trajectory HMM", which also uses (10) as the training criterion. Instead of solving (6), the parameter set $\boldsymbol{\lambda}$ is thus trained by solving

$$\{\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{s}}\} = \underset{\boldsymbol{\lambda},\boldsymbol{s}}{\mathrm{argmax}}\ \log p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda})p(\boldsymbol{s}), \tag{13}$$

where $\boldsymbol{c}$ is treated as the observed data.

Unlike the regular HMM, the conditional independence assumption on observed vectors does not hold in the trajectory HMM: While the regular HMM assumes that each observed vector depends only on the current state, (10) indicates that the observed vector $\boldsymbol{c}_t$ at each frame depends on the entire state sequence. This implies that it is difficult to directly apply the efficient decoding and training algorithms used in the HMM framework (such as the Viterbi algorithm and the Forward-Backward algorithm). Thus, some approximations and brute-force methods are usually necessary to perform training and decoding [3]. Because of this, the decoding algorithm is not guaranteed to find the optimal state sequence and the training algorithm is not guaranteed to converge to a local optimal solution. Note that this also applies to the minimum generation error (MGE) training framework [4], which uses (10) in which $\boldsymbol{V}_{\boldsymbol{s}}$ is replaced by an identity matrix as the training criterion.

3. LATENT TRAJECTORY HMM

3.1. Model

While the HMM is only capable of describing piecewise stationary sequences of data vectors, the trajectory HMM is capable of describing continuously varying sequences of data vectors, governed by discrete hidden states. This feature is notable in that it can be used to model many kinds of time series data that are continuous in nature, however, it causes a difficulty as regards parameter inference. We propose introducing a conceptually similar framework based on a different formulation, which is advantageous in that it alleviates the difficulty related to parameter inference.

Instead of treating $\boldsymbol{o}$ as a function of $\boldsymbol{c}$, we treat $\boldsymbol{o}$ as a latent variable that is related to $\boldsymbol{c}$ through a soft constraint $\boldsymbol{o} \simeq \boldsymbol{W}\boldsymbol{c}$. The relationship $\boldsymbol{o} \simeq \boldsymbol{W}\boldsymbol{c}$ can be expressed through the conditional distribution $p(\boldsymbol{c}|\boldsymbol{o})$. For example, we can define $p(\boldsymbol{c}|\boldsymbol{o})$ as

$$p(\boldsymbol{c}|\boldsymbol{o}) \propto \exp\left\{-\frac{1}{2}(\boldsymbol{W}\boldsymbol{c}-\boldsymbol{o})^{\mathsf{T}}\boldsymbol{\Lambda}(\boldsymbol{W}\boldsymbol{c}-\boldsymbol{o})\right\}, \qquad (14)$$

where $\boldsymbol{\Lambda}$ is a constant positive definite matrix that can be set arbitrarily. Indeed, this probability density function becomes larger as $\boldsymbol{o}$ approaches $\boldsymbol{W}\boldsymbol{c}$. By completing the square in the exponent of (14) with respect to c, we can write $p(\boldsymbol{c}|\boldsymbol{o})$ explicitly as

$$p(\boldsymbol{c}|\boldsymbol{o}) = \mathcal{N}(\boldsymbol{c}; \boldsymbol{m}_{c|o}, \boldsymbol{\Lambda}_{c|o}^{-1}), \qquad (15)$$

where

$$\boldsymbol{m}_{c|o} = \boldsymbol{H}\boldsymbol{o}, \qquad (16)$$

$$\boldsymbol{H} = (\boldsymbol{W}^{\mathsf{T}}\boldsymbol{\Lambda}\boldsymbol{W})^{-1}\boldsymbol{W}^{\mathsf{T}}\boldsymbol{\Lambda}, \qquad (17)$$

$$\boldsymbol{\Lambda}_{c|o} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{\Lambda}\boldsymbol{W}. \qquad (18)$$

By using this and $p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})$ defined in (2), we can write $p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda})$ as

$$p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda}) = \int p(\boldsymbol{c}|\boldsymbol{o})p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})\mathrm{d}\boldsymbol{o}, \qquad (19)$$

in a different way from (10). Geometrically, this can be viewed as a marginal distribution of the set of the projected values of $\boldsymbol{o}$ onto the subspace $\boldsymbol{o} = \boldsymbol{W}\boldsymbol{c}$. From (2) and (15), the joint likelihood $p(\boldsymbol{c},\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})$ can be written as

$$\begin{aligned} &p(\boldsymbol{c},\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda}) = p(\boldsymbol{c}|\boldsymbol{o})p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda}) \\ &\propto \exp\left\{-\frac{1}{2}\left(\begin{bmatrix}\boldsymbol{c}\\ \boldsymbol{o}\end{bmatrix}-\boldsymbol{m}_x\right)^{\mathsf{T}}\boldsymbol{\Lambda}_x\left(\begin{bmatrix}\boldsymbol{c}\\ \boldsymbol{o}\end{bmatrix}-\boldsymbol{m}_x\right)\right\}, \qquad (20)\end{aligned}$$

where

$$\boldsymbol{m}_x = \boldsymbol{\Lambda}_x^{-1}\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{\mu}_{\boldsymbol{s}}\end{bmatrix}, \qquad (21)$$

$$\boldsymbol{\Lambda}_x = \begin{bmatrix}\boldsymbol{\Lambda}_{c|o} & -\boldsymbol{W}^{\mathsf{T}}\boldsymbol{\Lambda}\\ -\boldsymbol{\Lambda}\boldsymbol{W} & \boldsymbol{H}^{\mathsf{T}}\boldsymbol{\Lambda}_{c|o}\boldsymbol{H}+\boldsymbol{U}_{\boldsymbol{s}}^{-1}\end{bmatrix}. \qquad (22)$$

Thus, $p(\boldsymbol{x}|\boldsymbol{s},\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_x, \boldsymbol{\Lambda}_x^{-1})$ where $\boldsymbol{x} = [\boldsymbol{c}^{\mathsf{T}}, \boldsymbol{o}^{\mathsf{T}}]^{\mathsf{T}}$. By using the blockwise matrix inversion formula, $\boldsymbol{\Lambda}_x^{-1}$ is given as

$$\boldsymbol{\Sigma}_x = \boldsymbol{\Lambda}_x^{-1} = \begin{bmatrix}\boldsymbol{\Sigma}_{cc} & \boldsymbol{\Sigma}_{co}\\ \boldsymbol{\Sigma}_{oc} & \boldsymbol{\Sigma}_{oo}\end{bmatrix}, \qquad (23)$$

where

$$\boldsymbol{\Sigma}_{cc} = \boldsymbol{\Lambda}_{c|o}^{-1} + \boldsymbol{H}\boldsymbol{U}_{\boldsymbol{s}}\boldsymbol{H}^{\mathsf{T}}, \qquad (24)$$

$$\boldsymbol{\Sigma}_{co} = \boldsymbol{H}\boldsymbol{U}_{\boldsymbol{s}}, \qquad (25)$$

$$\boldsymbol{\Sigma}_{oc} = \boldsymbol{U}_{\boldsymbol{s}}\boldsymbol{H}^{\mathsf{T}}, \qquad (26)$$

$$\boldsymbol{\Sigma}_{oo} = \boldsymbol{U}_{\boldsymbol{s}}. \qquad (27)$$

Hence, (19) can be written as

$$p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{c}; \boldsymbol{H}\boldsymbol{\mu}_{\boldsymbol{s}}, \boldsymbol{\Sigma}_{cc}). \qquad (28)$$

We call this model the “latent trajectory HMM.” With this framework, given a state sequence $\boldsymbol{s}$ and a parameter set $\boldsymbol{\lambda}$, $\boldsymbol{c}$ is generated according to

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}}{\mathrm{argmax}}\, p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda}). \qquad (29)$$

Obviously, the solution to this is $\boldsymbol{H}\boldsymbol{\mu}_{\boldsymbol{s}}$.

It is important to note that with this framework, the parameter inference problem can be dealt with using the Expectation-Maximization (EM) algorithm by treating the joint vector $[\boldsymbol{c}^{\mathsf{T}}, \boldsymbol{o}^{\mathsf{T}}]^{\mathsf{T}}$ as the complete data.

3.2. Decoding and training algorithms

As with the trajectory HMM framework, the present framework uses $p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda})$ for feature sequence generation, state decoding and parameter training in a consistent manner. The problems of state decoding and parameter training can be formulated as the following optimization problems:

$$\hat{\boldsymbol{s}} = \underset{\boldsymbol{s}}{\mathrm{argmax}}\ \log p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda})p(\boldsymbol{s}) \qquad (30)$$

$$\{\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{s}}\} = \underset{\boldsymbol{\lambda},\boldsymbol{s}}{\mathrm{argmax}}\ \log p(\boldsymbol{c}|\boldsymbol{s},\boldsymbol{\lambda})p(\boldsymbol{s}). \qquad (31)$$

Since the decoding problem (30) is a subproblem of the training problem (31), here we only derive an algorithm for solving the training problem (31).

By regarding the set consisting of $\boldsymbol{c}$ and $\boldsymbol{o}$ as the complete data, this problem can be viewed as an incomplete data problem, which can be dealt with using the Expectation-Maximization (EM) algorithm. The likelihood of $\boldsymbol{s}$ and $\boldsymbol{\lambda}$ given the complete data is given by (20). By taking the conditional expectation of $\log p(\boldsymbol{c},\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})$ with respect to $\boldsymbol{o}$ given $\boldsymbol{c}$, $\boldsymbol{s} = \boldsymbol{s}'$ and $\boldsymbol{\lambda} = \boldsymbol{\lambda}'$, and then adding $\log p(\boldsymbol{q})$, we obtain an auxiliary function

$$Q(\boldsymbol{s},\boldsymbol{\lambda}) := \mathbb{E}_{\boldsymbol{o}|\boldsymbol{c},\boldsymbol{s}',\boldsymbol{\lambda}'}[\log p(\boldsymbol{c},\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})] + \log p(\boldsymbol{s}). \qquad (32)$$

By leaving only the terms that depend on $\boldsymbol{s}$ and $\boldsymbol{\lambda}$, $Q(\boldsymbol{s},\boldsymbol{\lambda})$ can be written as

$$Q(\boldsymbol{s},\boldsymbol{\lambda}) \overset{\boldsymbol{s},\boldsymbol{\lambda}}{=} \mathbb{E}_{\boldsymbol{o}|\boldsymbol{c},\boldsymbol{s}',\boldsymbol{\lambda}'}[\log p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})] + \log p(\boldsymbol{s})$$

$$= -\frac{1}{2}\big\{\log|\boldsymbol{U}_{\boldsymbol{s}}| + \mathrm{Tr}\big(\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{R}\big) \\ - 2\boldsymbol{\mu}_{\boldsymbol{s}}^{\mathsf{T}}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\bar{\boldsymbol{o}} + \boldsymbol{\mu}_{\boldsymbol{s}}^{\mathsf{T}}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{\mu}_{\boldsymbol{s}}\big\} + \log p(\boldsymbol{s}), \quad (33)$$

where

$$\begin{aligned}\bar{\boldsymbol{o}} &= \mathbb{E}_{\boldsymbol{o}|\boldsymbol{c},\boldsymbol{s}',\boldsymbol{\lambda}'}[\boldsymbol{o}] \\ &= \boldsymbol{\mu}'_{\boldsymbol{s}'} + \boldsymbol{\Sigma}'_{oc}\boldsymbol{\Sigma}'^{-1}_{cc}(\boldsymbol{c} - \boldsymbol{H}\boldsymbol{\mu}'_{\boldsymbol{s}'}), & (34)\\ \boldsymbol{R} &= \mathbb{E}_{\boldsymbol{o}|\boldsymbol{c},\boldsymbol{s}',\boldsymbol{\lambda}'}[\boldsymbol{o}\boldsymbol{o}^{\mathsf{T}}] \\ &= \boldsymbol{\Sigma}'_{oo} - \boldsymbol{\Sigma}'_{oc}\boldsymbol{\Sigma}'^{-1}_{cc}\boldsymbol{\Sigma}'_{co} + \bar{\boldsymbol{o}}\bar{\boldsymbol{o}}^{\mathsf{T}}. & (35)\end{aligned}$$

Here, the prime mark indicates the values obtained using the model parameters updated at the previous iteration. Since $\boldsymbol{U}_{\boldsymbol{s}}$ is a block diagonal matrix, as given in (5), (33) can be decomposed into the sum of T individual terms:

$$Q(\boldsymbol{s},\boldsymbol{\lambda}) \stackrel{\boldsymbol{s},\boldsymbol{\lambda}}{=} -\frac{1}{2}\sum_{t=1}^{T}\{\log|\boldsymbol{U}_{s_t}| + \mathrm{Tr}[\boldsymbol{U}_{s_t}^{-1}\boldsymbol{R}_t] \\ - 2\boldsymbol{\mu}_{s_t}^{\mathsf{T}}\boldsymbol{U}_{s_t}^{-1}\bar{\boldsymbol{o}}_t + \boldsymbol{\mu}_{s_t}^{\mathsf{T}}\boldsymbol{U}_{s_t}^{-1}\boldsymbol{\mu}_{s_t}\} \\ + \log\pi_{s_1} + \sum_{t=2}^{T}\log\pi_{s_{t-1},s_t}, \quad (36)$$

where

$$\bar{\boldsymbol{o}} = \begin{bmatrix}\bar{\boldsymbol{o}}_1\\ \vdots \\ \bar{\boldsymbol{o}}_T\end{bmatrix}, \quad \boldsymbol{R} = \begin{bmatrix}\boldsymbol{R}_1 & & * \\ & \ddots & \\ * & & \boldsymbol{R}_T\end{bmatrix}. \quad (37)$$

With fixed $\boldsymbol{\lambda}$, $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{\lambda})$ can be maximized with respect to $\boldsymbol{s}$ by employing the Viterbi algorithm. With fixed $\boldsymbol{s}$, $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{\lambda})$ is maximized with respect to $\boldsymbol{\lambda}$ when

$$\boldsymbol{\mu}_i = \frac{\sum_t \mathbf{1}[s_t = i]\bar{\boldsymbol{o}}_t}{\sum_t \mathbf{1}[s_t = i]}, \quad (38)$$

$$\boldsymbol{U}_i = \frac{\sum_t \mathbf{1}[s_t = i](\boldsymbol{R}_t - \bar{\boldsymbol{o}}_t\boldsymbol{\mu}_i^{\mathsf{T}} - \boldsymbol{\mu}_i\bar{\boldsymbol{o}}_t^{\mathsf{T}} + \boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathsf{T}})}{\sum_t \mathbf{1}[s_t = i]}, \quad (39)$$

$$\pi_{i,j} = \frac{\sum_t \mathbf{1}[s_{t-1} = i, s_t = j]}{\sum_t \mathbf{1}[s_{t-1} = i]}, \quad (40)$$

if $\sum_t \mathbf{1}[s_t = i] \neq 0$ where i and j denote state indices and $\mathbf{1}[\cdot]$ denotes an indicator function that takes the value 1 if its argument is true and 0 otherwise.

Overall, the parameter training algorithm can be summarized as follows:

(E-step) Substitute $\boldsymbol{s}$ and $\boldsymbol{\lambda}$ into $\boldsymbol{s}'$ and $\boldsymbol{\lambda}'$ and recompute $\bar{\boldsymbol{o}}$ and $\boldsymbol{R}$ using (34) and (35).

(M-step) Update $\boldsymbol{\lambda}$ using (38)–(40) and find

$$\boldsymbol{s} = \operatorname*{argmax}_{\boldsymbol{s}} Q(\boldsymbol{s},\boldsymbol{\lambda}) \quad (41)$$

using the Viterbi algorithm.

Note that if $\boldsymbol{\lambda}$ is fixed, the above algorithm reduces to a state decoding algorithm.

It may appear that a huge amount of computation for inverting $\boldsymbol{\Sigma}'_{cc}$ is required to compute $\bar{\boldsymbol{o}}$ and $\boldsymbol{R}$. However, this can be carried out very efficiently. First, by using the Woodbury matrix identity, $\boldsymbol{\Sigma}'^{-1}_{cc}$ can be written as $\boldsymbol{\Lambda}_{c|o} - \boldsymbol{\Lambda}_{c|o}((\boldsymbol{H}\boldsymbol{U}_{\boldsymbol{s}}\boldsymbol{H}^{\mathsf{T}})^{-1} + \boldsymbol{\Lambda}_{c|o})^{-1}\boldsymbol{\Lambda}_{c|o}$. Next, since $\boldsymbol{\Lambda}$ can be set arbitrarily, we set $\boldsymbol{\Lambda}$ at $\boldsymbol{U}_{\boldsymbol{s}}^{-1}$ to compute $(\boldsymbol{H}\boldsymbol{U}_{\boldsymbol{s}}\boldsymbol{H}^{\mathsf{T}})^{-1}$. Under this setting, $(\boldsymbol{H}\boldsymbol{U}_{\boldsymbol{s}}\boldsymbol{H}^{\mathsf{T}})^{-1}$ is given as $\boldsymbol{W}^{\mathsf{T}}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{W}$. Since both $\boldsymbol{W}^{\mathsf{T}}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{W}$ and $\boldsymbol{\Lambda}_{c|o}$ are sparse symmetric band matrices, $(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{U}_{\boldsymbol{s}}^{-1}\boldsymbol{W} + \boldsymbol{\Lambda}_{c|o})^{-1}\boldsymbol{\Lambda}_{c|o}$ can be computed efficiently using the Cholesky decomposition.

To initialize $\boldsymbol{s}$, one reasonable way would be to search for $\boldsymbol{s} = \operatorname{argmax}_{\boldsymbol{s}} p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})$ using the Viterbi algorithm.

3.3. Variational learning algorithm

We describe a different approach for parameter training based on variational inference. The random variables of interest in our model are $\boldsymbol{o}$, $\boldsymbol{s}$ and $\boldsymbol{\lambda} = \{\boldsymbol{\mu}, \boldsymbol{P}, \boldsymbol{\pi}\}$ where $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}_{1\le i\le I}$, $\boldsymbol{P} = \{\boldsymbol{P}_i := \boldsymbol{U}_i^{-1}\}_{1\le i\le I}$, and $\boldsymbol{\pi} = \{\boldsymbol{\pi}_{i,j}\}_{1\le i\le I, 1\le j\le J}$. We denote the entire set of the above parameters as $\boldsymbol{\Theta} = \{\boldsymbol{o},\boldsymbol{s},\boldsymbol{\lambda}\}$. Our goal is to compute the posterior

$$p(\boldsymbol{\Theta}|\boldsymbol{c}) = \frac{p(\boldsymbol{c},\boldsymbol{\Theta})}{p(\boldsymbol{c})}. \quad (42)$$

By using the conditional distributions defined in 3.1, we can write the joint distribution $p(\boldsymbol{c},\boldsymbol{\Theta})$ as

$$p(\boldsymbol{c},\boldsymbol{\Theta}) = p(\boldsymbol{c}|\boldsymbol{o})p(\boldsymbol{o}|\boldsymbol{s},\boldsymbol{\lambda})p(\boldsymbol{s}). \quad (43)$$

To obtain the exact posterior $p(\boldsymbol{\Theta}|\boldsymbol{c})$, we must compute $p(\boldsymbol{c})$, which involves many intractable integrals. Instead of obtaining the exact posterior, we consider approximating this posterior variationally by solving an optimization problem:

$$\operatorname*{argmin}_{q} \mathrm{KL}(q(\boldsymbol{\Theta})\|p(\boldsymbol{\Theta}|\boldsymbol{c})), \quad (44)$$

where $\mathrm{KL}(\cdot\|\cdot)$ denotes the Kullback-Leibler (KL) divergence between its two arguments, i.e.,

$$\mathrm{KL}(q(\boldsymbol{\Theta})\|p(\boldsymbol{\Theta}|\boldsymbol{c})) \\ = \sum_{\boldsymbol{s}}\iint q(\boldsymbol{o},\boldsymbol{s},\boldsymbol{\lambda})\log\frac{q(\boldsymbol{o},\boldsymbol{s},\boldsymbol{\lambda})}{p(\boldsymbol{o},\boldsymbol{s},\boldsymbol{\lambda}|\boldsymbol{c})}\mathrm{d}\boldsymbol{o}\mathrm{d}\boldsymbol{\lambda}. \quad (45)$$

By restricting the class of the approximate distributions to those that factorize into independent factors:

$$q(\boldsymbol{o},\boldsymbol{s},\boldsymbol{\lambda}) = q(\boldsymbol{o})q(\boldsymbol{s})q(\boldsymbol{\mu},\boldsymbol{P})q(\boldsymbol{\pi}), \quad (46)$$

we can use a simple coordinate ascent algorithm to find a local optimum of (44). It can be shown using the calculus of variations that the “optimal” distribution for each of the factors can be expressed as:

$$\hat{q}(\boldsymbol{X}) \propto \exp \mathbb{E}_{\boldsymbol{\Theta}\setminus\boldsymbol{X}}[\log p(\boldsymbol{c},\boldsymbol{\Theta})], \tag{47}$$

where $\boldsymbol{X}$ indicates one of the factors and $\mathbb{E}_{\boldsymbol{\Theta}\setminus\boldsymbol{X}}[\log p(\boldsymbol{c},\boldsymbol{\Theta})]$ is the expectation of the joint probability of the data and latent variables, taken over all variables except $\boldsymbol{X}$. From (47), the variational distributions are given in the following form:

$$\hat{q}(\boldsymbol{o}) = \mathcal{N}(\boldsymbol{o};\boldsymbol{m},\boldsymbol{\Gamma}), \tag{48}$$

$$\hat{q}(\boldsymbol{\mu},\boldsymbol{P}) = \prod_i \mathcal{N}(\boldsymbol{\mu}_i;\boldsymbol{\rho_i},(\beta_i\boldsymbol{P}_i)^{-1})\mathcal{W}(\boldsymbol{P}_i;\boldsymbol{B}_i,\nu_i), \tag{49}$$

$$\hat{q}(\boldsymbol{\pi}_i) = \mathrm{Dir}(\boldsymbol{\pi}_i;\boldsymbol{\alpha}_i), \tag{50}$$

where the parameters are updated via the following equations

$$\boldsymbol{m} = \begin{bmatrix} \boldsymbol{m}_1 \\ \vdots \\ \boldsymbol{m}_T \end{bmatrix} \leftarrow \boldsymbol{R}^{-1}\boldsymbol{r}, \tag{51}$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\Gamma}_1 & & * \\ & \ddots & \\ * & & \boldsymbol{\Gamma}_T \end{bmatrix} \leftarrow \boldsymbol{R}^{-1}, \tag{52}$$

$$\boldsymbol{R} = \begin{bmatrix} \sum_i q(s_1\!=\!i)\nu_i\boldsymbol{B}_i & & O \\ & \ddots & \\ O & & \sum_i q(s_T\!=\!i)\nu_i\boldsymbol{B}_i \end{bmatrix} + \boldsymbol{H}^\mathsf{T}\boldsymbol{\Lambda}_{c|o}\boldsymbol{H}, \tag{53}$$

$$\boldsymbol{r} = \begin{bmatrix} \sum_i q(s_1\!=\!i)\nu_i\boldsymbol{B}_i\boldsymbol{\rho}_i \\ \vdots \\ \sum_i q(s_T\!=\!i)\nu_i\boldsymbol{B}_i\boldsymbol{\rho}_i \end{bmatrix} + \boldsymbol{H}^\mathsf{T}\boldsymbol{\Lambda}_{c|o}\boldsymbol{c}, \tag{54}$$

$$\beta_i \leftarrow \sum_t q(s_t\!=\!i), \tag{55}$$

$$\boldsymbol{\rho}_i \leftarrow \frac{1}{\beta_i}\sum_t q(s_t\!=\!i)\boldsymbol{m}_t, \tag{56}$$

$$\boldsymbol{B}_i^{-1} \leftarrow \sum_t q(s_t\!=\!i)(\boldsymbol{\Gamma}_t + \boldsymbol{m}_t\boldsymbol{m}_t^\mathsf{T}) - \beta_i\boldsymbol{\rho}_i\boldsymbol{\rho}_i^\mathsf{T}, \tag{57}$$

$$\nu_i \leftarrow \beta_i + 3D, \tag{58}$$

$$\alpha_{i,j} \leftarrow 1 + \sum_t q(s_{t-1}\!=\!i, s_t\!=\!j). \tag{59}$$

$\mathcal{W}$ and Dir denote the Wishart distribution and the Dirichlet distribution, respectively, defined as

$$\mathcal{W}(\boldsymbol{X};\boldsymbol{V},\nu) \propto |\boldsymbol{X}|^{\frac{\nu-d-1}{2}} e^{-\frac{1}{2}\mathrm{Tr}(\boldsymbol{V}^{-1}\boldsymbol{X})}, \tag{60}$$

$$\mathrm{Dir}(\boldsymbol{x};\boldsymbol{\alpha}) \propto \prod_i x_i^{\alpha_i-1}, \tag{61}$$

where $\boldsymbol{X}$ is a $d\times d$ symmetric matrix of random variables that is positive definite and $\boldsymbol{V}$ is a $d \times d$ positive definite matrix. $q(s_t = i)$ and $q(s_{t-1} = i, s_t = j)$ can be computed using the forward-backward algorithm as a subroutine, as in [12].

4. EXPERIMENTS

To confirm the generalization ability of the present model and the convergence speed of the present training algorithm, we conducted parameter training experiments using mel-cepstrum sequences of speech as experimental data. We chose the parameter training algorithm for the original trajectory HMM developed by Zen et al. as the baseline method. Zen’s algorithm uses the method of steepest ascent for updating $\boldsymbol{\lambda}$ and the “delayed decision Viterbi algorithm” for updating $\boldsymbol{s}$. Readers are referred to [3] for the details. Since the degrees of freedom of the trajectory HMM and the latent trajectory HMM are exactly the same when the numbers of hidden states are the same, the difference of the log-likelihood scores obtained with the present and baseline algorithms would reflect the difference in their generalization abilities. The experimental conditions were as follows. We used 25 speech data excerpted from the ATR speech database, from each of which we obtained the mel-cepstrum sequence of the first 250 frames. For both the proposed model and the trajectory HMM, the numbers of hidden states were set at 14. $\boldsymbol{\Lambda}$ were fixed at

$$\boldsymbol{\Lambda} = \mathrm{diag}(\underbrace{\boldsymbol{A},\ldots,\boldsymbol{A}}_{T}), \tag{62}$$

$$\boldsymbol{A} = \begin{bmatrix} 0.0001^{-1} & 0 & 0 \\ 0 & 0.01^{-1} & 0 \\ 0 & 0 & 0.01^{-1} \end{bmatrix}. \tag{63}$$

The workstation used to perform the experiments had an Intel Core i3-2120 Processor with a 3.3GHz ×4 clock speed and a 7.7GB memory.

Fig. 2 shows the evolution of the log-likelihoods with respect to the number of iterations and computation time during the parameter training of the proposed model and the conventional trajectory HMM. As Fig. 2 shows, the present algorithm converged faster than the conventional algorithm. This reveals the effectiveness of the combined use of efficient statistical inference techniques such as the EM algorithm and the dynamic programming principle by the proposed algorithm. It is also worth noting that the converged value of the log-likelihood obtained with the proposed algorithm was greater than that obtained with the conventional algorithm. This implies the possibility that, compared with the conventional model, the proposed model has a higher generalization ability, namely an ability to fit an arbitrary set of feature sequences, given that the degrees-of-freedom of the two models were the same.

Fig. 3 shows an example of parameter generation using the proposed model. After training $\boldsymbol{\lambda}$ using 25 speech data, a feature sequence $\hat{\boldsymbol{c}}$ was generated according to (29) given a state sequence $\boldsymbol{s}$. The figure at the top shows the spectrogram of a speech sample and the figure at the bottom shows the spectrogram constructed using $\hat{\boldsymbol{c}}$ obtained using the trained $\boldsymbol{\lambda}$ and the state sequence $\boldsymbol{s}$ labeled from the speech sample. As Fig. 3 shows, the proposed model was able to represent the continuously time-varying nature of speech spectrograms reasonably well, showing that it has a similar property to the trajectory HMM.

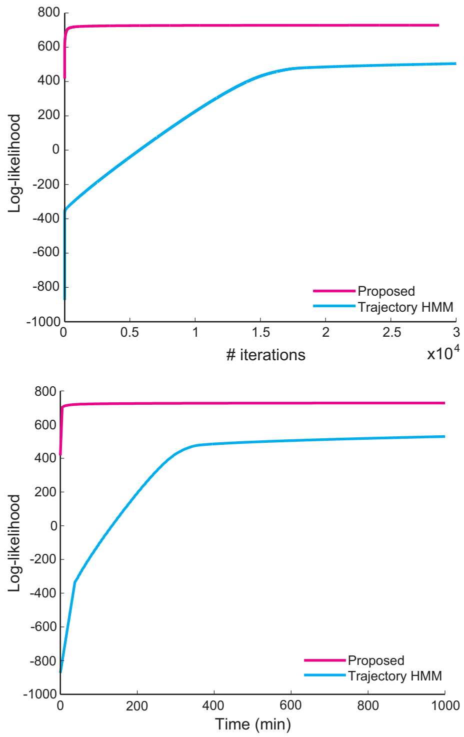

Fig. 2. Evolutions of the log-likelihoods with respect to the number of iterations (top) and the computation time (bottom).

5. CONCLUSIONS

Inspired by the trajectory HMM framework proposed by Zen et al., this paper proposed a probabilistic generative model for describing continuously time-varying sequences of data vectors governed by discrete hidden states. The proposed model is advantageous over the conventional trajectory HMM in that it makes it possible to derive convergence-guaranteed and efficient algorithms for parameter training and state decoding. Interesting future work involves incorporating the proposed model into the factorial HMM formulation to develop a new method for audio source separation that takes account of the dynamics of source spectra.

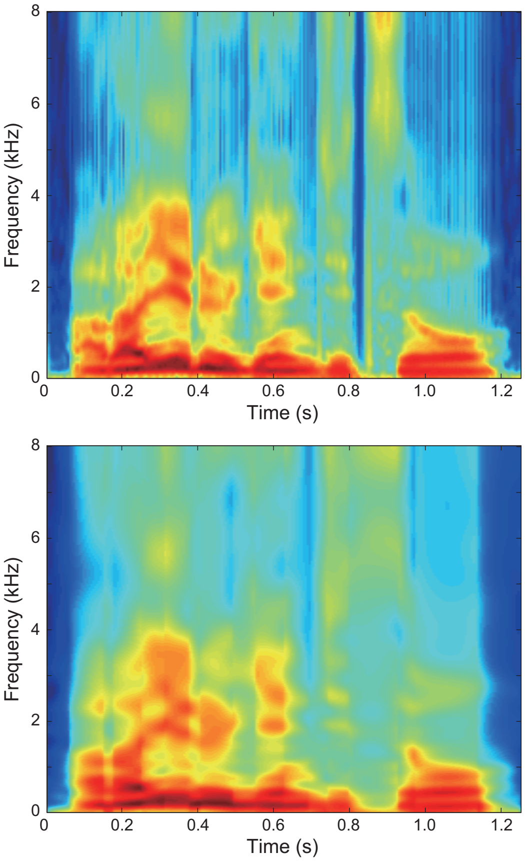

Fig. 3. Example of parameter generation using the proposed model. The spectrogram of a training sample (top) and the spectrogram constructed using the generated parameters (bottom).

6. REFERENCES

[1] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," in *Proc. The 6th European Conference on Speech Communication and Technology (EUROSPEECH 1999)*, 1999, pp. 2347–2350.

[2] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *Proc. 2000 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2000)*, 2000, pp. 1315–1318.

[3] H. Zen, K. Tokuda, and T. Kitamura, "Reformulating the HMM as a trajetory model by imposing explicit relationships between static and dynamic feature vector sequences," *Computer Speech and Language*, vol. 21, pp. 153–173, 2007.

[4] Y.-J. Wu and R.H. Wang, "Minimum generation error training for HMM-based speech synthesis," in *Proc. 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, 2006, pp. 89–92.

[5] T. Toda, A.W. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *IEEE Trasanctions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2222–2235, 2007.

[6] T. Toda, A.W. Black, and K. Tokuda, "Statistical mapping between articulatory movements and acoustic spectrum using a Gaussian mixture model," *Speech Communication*, vol. 50, no. 3, pp. 215–227, 2007.

[7] L. Zhang and S. Renals, "Acoustic-articulatory modelling with the trajectory HMM," *IEEE Signal Processing Letters*, vol. 15, pp. 245–248, 2008.

[8] M. Nakano, J. Le Roux, H. Kameoka, T. Nakamura, N. Ono, and S. Sagayama, "Bayesian nonparametric spectrogram modeling based on infinite factorial infinite hidden Markov model," in *Proc. 2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA2011)*, 2011, pp. 325–328.

[9] T. Higuchi, H. Takeda, T. Nakamura, and H. Kameoka, "A unified approach for underdetermined blind signal separation and source activity detection by multichannel factorial hidden markov models," in *Proc. The 15th Annual Conference of the International Speech Communication Association (Interspeech 2014)*, 2014, pp. 850–854.

[10] T. Higuchi and H. Kameoka, "Joint audio source separation and dereverberation based on multichannel factorial hidden markov model," in *Proc. The 24th IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2014)*, 2014.

[11] T. Higuchi and H. Kameoka, "Unified approach for underdetermined BSS, VAD, dereverberation and DOA estimation with multichannel factorial HMM," in *Proc. of The 2nd IEEE Global Conference on Signal and Information Processing (GlobalSIP 2014)*, 2014.

[12] M.J. Beal, *Variational Algorithms for Approximate Bayesian Inference*, Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London, 1998.