Journal of Machine Learning Research 15 (2014) 697-747 Submitted 10/12; Revised 6/13; Published 2/14

Clustering Hidden Markov Models with Variational HEM

Emanuele Coviello ECOVIELL@UCSD.EDU
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093, USA

Antoni B. Chan ABCHAN@CITYU.EDU.HK
Department of Computer Science
City University of Hong Kong
Kowloon Tong, Hong Kong

Gert R.G. Lanckriet GERT@ECE.UCSD.EDU
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093, USA

Editor: Tony Jebara

Abstract

The hidden Markov model (HMM) is a widely-used generative model that copes with sequential data, assuming that each observation is conditioned on the state of a hidden Markov chain. In this paper, we derive a novel algorithm to cluster HMMs based on the hierarchical EM (HEM) algorithm. The proposed algorithm i) clusters a given collection of HMMs into groups of HMMs that are similar, in terms of the distributions they represent, and ii) characterizes each group by a "cluster center", that is, a novel HMM that is representative for the group, in a manner that is consistent with the underlying generative model of the HMM. To cope with intractable inference in the E-step, the HEM algorithm is formulated as a variational optimization problem, and efficiently solved for the HMM case by leveraging an appropriate variational approximation. The benefits of the proposed algorithm, which we call variational HEM (VHEM), are demonstrated on several tasks involving time-series data, such as hierarchical clustering of motion capture sequences, and automatic annotation and retrieval of music and of online hand-writing data, showing improvements over current methods. In particular, our variational HEM algorithm effectively leverages large amounts of data when learning annotation models by using an efficient hierarchical estimation procedure, which reduces learning times and memory requirements, while improving model robustness through better regularization.

Keywords: Hierarchical EM algorithm, clustering, hidden Markov model, hidden Markov mixture model, variational approximation, time-series classification

1. Introduction

The hidden Markov model (HMM) (Rabiner and Juang, 1993) is a probabilistic model that assumes a signal is generated by a double embedded stochastic process. A discrete-time hidden state process, which evolves as a Markov chain, encodes the dynamics of the signal, while an observation process encodes the appearance of the signal at each time,

conditioned on the current state. HMMs have been successfully applied to a variety of fields, including speech recognition (Rabiner and Juang, 1993), music analysis (Qi et al., 2007) and identification (Batlle et al., 2002), online hand-writing recognition (Nag et al., 1986), analysis of biological sequences (Krogh et al., 1994), and clustering of time series data (Jebara et al., 2007; Smyth, 1997; Alon et al., 2003).

This paper is about clustering HMMs. More precisely, we are interested in an algorithm that, given a collection of HMMs, partitions them into K clusters of "similar" HMMs, while also learning a representative HMM "cluster center" that concisely and appropriately represents each cluster. This is similar to standard k-means clustering, except that the data points are HMMs now instead of vectors in $\mathbb{R}^d$.

Various applications motivate the design of HMM clustering algorithms, ranging from hierarchical clustering of sequential data (e.g., speech or motion sequences modeled by HMMs as by Jebara et al. 2007), to hierarchical indexing for fast retrieval, to reducing the computational complexity of estimating mixtures of HMMs from large data sets (e.g., semantic annotation models for music and video)—by clustering HMMs, efficiently estimated from many small subsets of the data, into a more compact mixture model of all data. However, there has been little work on HMM clustering and, therefore, its applications.

Existing approaches to clustering HMMs operate directly on the HMM *parameter* space, by grouping HMMs according to a suitable pairwise distance defined in terms of the HMM parameters. However, as HMM parameters lie on a non-linear manifold, a simple application of the k-means algorithm will not succeed in the task, since it assumes real vectors in a Euclidean space. In addition, such an approach would have the additional complication that HMM parameters for a particular generative model are not unique, that is, a permutation of the states leads to the same generative model. One solution, proposed by Jebara et al. (2007), first constructs an appropriate similarity matrix between all HMMs that are to be clustered (e.g., based on the Bhattacharya affinity, which depends non-linearly on the HMM parameters Jebara et al. 2004; Hershey and Olsen 2008), and then applies spectral clustering. While this approach has proven successful to group HMMs into similar clusters (Jebara et al., 2007), it does not directly address the issue of generating HMM cluster centers. Each cluster can still be represented by choosing one of the given HMMs, for example, the HMM which the spectral clustering procedure maps the closest to each spectral clustering center. However, this may be suboptimal for some applications of HMM clustering, for example in hierarchical estimation of annotation models. Another distance between HMM distributions suitable for spectral clustering is the KL divergence, which in practice has been approximated by sampling sequences from one model and computing their log-likelihood under the other (Juang and Rabiner, 1985; Zhong and Ghosh, 2003; Yin and Yang, 2005).

Instead, in this paper we propose to cluster HMMs *directly* with respect to the *probability distributions* they represent. The probability distributions of the HMMs are used throughout the whole clustering algorithm, and not only to construct an initial embedding as Jebara et al. (2007). By clustering the output distributions of the HMMs, marginalized over the hidden-state distributions, we avoid the issue of multiple equivalent parameterizations of the hidden states. We derive a hierarchical expectation maximization (HEM) algorithm that, starting from a collection of input HMMs, estimates a smaller mixture model of HMMs that concisely represents and clusters the input HMMs (i.e., the input HMM distributions guide the estimation of the output mixture distribution).

The HEM algorithm is a *generalization* of the EM algorithm—the EM algorithm can be considered as a special case of HEM for a mixture of delta functions as input. The main difference between HEM and EM is in the E-step. While the EM algorithm computes the sufficient statistics given the observed data, the HEM algorithm calculates the expected sufficient statistics averaged over all possible observations generated by the input probability models. Historically, the first HEM algorithm was designed to cluster *Gaussian* probability distributions (Vasconcelos and Lippman, 1998). This algorithm starts from a Gaussian mixture model (GMM) with $K^{(b)}$ components and reduces it to another GMM with fewer components, where each of the mixture components of the reduced GMM represents, that is, *clusters*, a group of the original Gaussian mixture components. More recently, Chan et al. (2010b) derived an HEM algorithm to cluster *dynamic texture* (DT) models (i.e., linear dynamical systems, LDSs) through their probability distributions. HEM has been applied successfully to construct GMM hierarchies for efficient image indexing (Vasconcelos, 2001), to cluster video represented by DTs (Chan et al., 2010a), and to estimate GMMs or DT mixtures (DTMs, that is, LDS mixtures) from large data sets for semantic annotation of images (Carneiro et al., 2007), video (Chan et al., 2010a) and music (Turnbull et al., 2008; Coviello et al., 2011). Note that HMMs cannot be clustered by using the original HEM by Vasconcelos and Lippman (1998). Specifically, the original formulation of HEM was designed for clustering data points represented by *individual* Gaussian models. When clustering HMMs, we are interested in assigning every HMM as a whole to a cluster, and do not want to treat their individual Gaussian states independently. Even with GMMs (as opposed to single Gaussians) this is not possible in closed form, since it would need the expected log likelihood of a mixture.

To extend the HEM framework from clustering Gaussians to clustering HMMs, additional marginalization over the hidden-state processes is required, as with DTs. However, while Gaussians and DTs allow tractable inference in the E-step of HEM, this is no longer the case for HMMs. Therefore, in this work, we derive a variational formulation of the HEM algorithm (VHEM), and then leverage a variational *approximation* derived by Hershey et al. (2007) (which has not been used in a learning context so far) to make the inference in the E-step tractable. The resulting algorithm not only clusters HMMs, but also learns novel HMMs that are representative centers of each cluster. The resulting VHEM algorithm can be generalized to handle other classes of graphical models, for which exact computation of the E-step in the standard HEM would be intractable, by leveraging similar variational approximations—for example, any mixtures of continuous exponential family distributions (e.g., Gaussian) the more general case of HMMs with emission probabilities that are (mixtures of) continuous exponential family distributions.

Compared to the spectral clustering algorithm of Jebara et al. (2007), the VHEM algorithm has several advantages that make it suitable for a variety of applications. First, the VHEM algorithm is capable of both clustering, as well as learning *novel* cluster centers, in a manner that is consistent with the underlying generative probabilistic framework. In addition, since it does not require sampling steps, it is also scalable with low memory requirements. As a consequence, VHEM for HMMs allows for efficient estimation of HMM mixtures from large data sets using a hierarchical estimation procedure. In particular, intermediate HMM mixtures are first estimated in *parallel* by running the EM algorithm on small independent portions of the data set. The final model is then estimated from the

intermediate models using the VHEM algorithm. Because VHEM is based on maximum-likelihood principles, it drives model estimation towards similar optimal parameter values as performing maximum-likelihood estimation on the full data set. In addition, by averaging over all possible observations compatible with the input models in the E-step, VHEM provides an implicit form of regularization that prevents over-fitting and improves robustness of the learned models, compared to a direct application of the EM algorithm on the full data set. Note that, in contrast to Jebara et al. (2007), VHEM does not construct a kernel embedding, and is therefore expected to be more efficient, especially for large $K^{(b)}$.

In summary, the contributions of this paper are three-fold: i) we derive a variational formulation of the HEM algorithm for clustering HMMs, which generates novel HMM centers representative of each cluster; ii) we evaluate VHEM on a variety of clustering, annotation, and retrieval problems involving time-series data, showing improvement over current clustering methods; iii) we demonstrate in experiments that VHEM can effectively learn HMMs from large sets of data, more efficiently than standard EM, while improving model robustness through better regularization. With respect to our previous work, the VHEM algorithm for HMMs was originally proposed by Coviello et al. (2012a)

The remainder of the paper is organized as follows. We review the hidden Markov model (HMM) and the hidden Markov mixture model (H3M) in Section 2. We present the derivation of the VHEM-H3M algorithm in Section 3, followed by a discussion in Section 4.boldsymbol Finally, we present experimental results in Sections 5 and 6.

2. The Hidden Markov (Mixture) Model

A hidden Markov model (HMM) $\mathcal{M}$ assumes a sequence of τ observations $\mathbf{y} = \{y_1, \ldots, y_\tau\}$ is generated by a double embedded stochastic process, where each observation (or emission) y_t at time t depends on the state of a discrete hidden variable x_t, and the sequence of hidden states $\mathbf{x} = \{x_1, \ldots, x_\tau\}$ evolves as a first-order Markov chain. The hidden variables can take one of S values, $\{1, \ldots, S\}$, and the evolution of the hidden process is encoded in a state transition matrix $A = [a_{\beta,\beta'}]_{\beta,\beta'=1,\ldots,S}$, where each entry, $a_{\beta,\beta'} = p(x_{t+1} = \beta' | x_t = \beta, \mathcal{M})$, is the probability of transitioning from state β to state β', and an initial state distribution $\pi = [\pi_1, \ldots, \pi_S]$, where $\pi_\beta = p(x_1 = \beta | \mathcal{M})$.

Each state β generates observations according to an emission probability density function, $p(y_t | x_t = \beta, \mathcal{M})$. Here, we assume the emission density is *time-invariant*, and modeled as a Gaussian mixture model (GMM) with M components:

$$p(y | x = \beta, \mathcal{M}) = \sum_{m=1}^{M} c_{\beta,m} p(y | \zeta = m, x = \beta, \mathcal{M}), \tag{1}$$

where $\zeta \sim \text{multinomial}(c_{\beta,1}, \ldots, c_{\beta,M})$ is the hidden assignment variable that selects the mixture component, with $c_{\beta,m}$ as the mixture weight of the mth component, and each component is a multivariate Gaussian distribution,

$$p(y | \zeta = m, x = \beta, \mathcal{M}) = \mathcal{N}(y; \mu_{\beta,m}, \Sigma_{\beta,m}),$$

with mean $\mu_{\beta,m}$ and covariance matrix $\Sigma_{\beta,m}$. The HMM is specified by the parameters

$$\mathcal{M} = \{\pi, A, \{\{c_{\beta,m}, \mu_{\beta,m}, \Sigma_{\beta,m}\}_{m=1}^{M}\}_{\beta=1}^{S}\},$$

which can be efficiently learned from an observation sequence $\mathbf{y}$ with the Baum-Welch algorithm (Rabiner and Juang, 1993), which is based on maximum likelihood estimation.

The probability distribution of a state sequence $\mathbf{x}$ generated by an HMM $\mathcal{M}$ is

$$p(\mathbf{x}|\mathcal{M}) = p(x_1|\mathcal{M}) \prod_{t=2}^{\tau} p(x_t|x_{t-1}, \mathcal{M}) = \pi_{x_1} \prod_{t=2}^{\tau} a_{x_{t-1},x_t},$$

while the joint likelihood of an observation sequence $\mathbf{y}$ and a state sequence $\mathbf{x}$ is

$$p(\mathbf{y}, \mathbf{x}|\mathcal{M}) = p(\mathbf{y}|\mathbf{x}, \mathcal{M}) p(\mathbf{x}|\mathcal{M}) = p(x_1|\mathcal{M}) \prod_{t=2}^{\tau} p(x_t|x_{t-1}, \mathcal{M}) \prod_{t=1}^{\tau} p(y_t|x_t, \mathcal{M}).$$

Finally, the observation likelihood of $\mathbf{y}$ is obtained by marginalizing out the state sequence from the joint likelihood,

$$p(\mathbf{y}|\mathcal{M}) = \sum_{\mathbf{x}} p(\mathbf{y}, \mathbf{x}|\mathcal{M}) = \sum_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}, \mathcal{M}) p(\mathbf{x}|\mathcal{M}), \tag{2}$$

where the summation is over all state sequences of length τ, and can be performed efficiently using the *forward algorithm* (Rabiner and Juang, 1993).

A hidden Markov mixture model (H3M) (Smyth, 1997) models a set of observation sequences as samples from a group of K hidden Markov models, each associated to a specific sub-behavior. For a given sequence, an assignment variable $z \sim \text{multinomial}(\omega_1, \cdots, \omega_K)$ selects the parameters of one of the K HMMs, where the kth HMM is selected with probability ω_k. Each mixture component is parametrized by

$$\mathcal{M}_z = \{\pi^z, A^z, \{\{c^z_{\beta,m}, \mu^z_{\beta,m}, \Sigma^z_{\beta,m}\}_{m=1}^{M}\}_{\beta=1}^{S}\},$$

and the H3M is parametrized by $\mathcal{M} = \{\omega_z, \mathcal{M}_z\}_{z=1}^{K}$, which can be estimated from a collection of observation sequences using the EM algorithm (Smyth, 1997; Alon et al., 2003).

To reduce clutter, here we assume that all the HMMs have the same number S of hidden states and that all emission probabilities have M mixture components. Our derivation could be easily extended to the more general case though.

3. Clustering Hidden Markov Models

Algorithms for clustering HMMs can serve a wide range of applications, from hierarchical clustering of sequential data (e.g., speech or motion sequences modeled by HMMs (Jebara et al., 2007)), to hierarchical indexing for fast retrieval, to reducing the computational complexity of estimating mixtures of HMMs from large weakly-annotated data sets—by clustering HMMs, efficiently estimated from many small subsets of the data, into a more compact mixture model of all data.

In this work we derive a hierarchical EM algorithm for clustering HMMs (HEM-H3M) with respect to their probability distributions. We approach the problem of clustering HMMs as reducing an input HMM mixture with a large number of components to a new mixture with fewer components. Note that different HMMs in the input mixture are allowed to have different weights (i.e., the mixture weights $\{\omega_z\}_{z=1}^{K}$ are not necessarily all equal).

One method for estimating the reduced mixture model is to generate samples from the input mixture, and then perform maximum likelihood estimation, that is, maximize the log-likelihood of these samples. However, to avoid explicitly generating these samples, we instead maximize the *expectation* of the log-likelihood with respect to the input mixture model, thus averaging over all possible samples from the input mixture model. In this way, the dependency on the samples is replaced by a marginalization with respect to the input mixture model. While such marginalization is tractable for Gaussians and DTs, this is no longer the case for HMMs. Therefore, in this work, we i) derive a variational formulation of the HEM algorithm (VHEM), and ii) specialize it to the HMM case by leveraging a variational approximation proposed by Hershey et al. (2007). Note that the work of Hershey et al. (2007) was proposed as an alternative to MCMC sampling for the computation of the KL divergence between two HMMs, and has not been used in a learning context so far.

We present the problem formulation in Section 3.1, and derive the algorithm in Sections 3.2, 3.3 and 3.4.

3.1 Formulation

Let $\mathcal{M}^{(b)}$ be a base hidden Markov mixture model with $K^{(b)}$ components. The goal of the VHEM algorithm is to find a reduced hidden Markov mixture model $\mathcal{M}^{(r)}$ with $K^{(r)} < K^{(b)}$ (i.e., fewer) components that represents $\mathcal{M}^{(b)}$ well. The likelihood of a random sequence $\mathbf{y} \sim \mathcal{M}^{(b)}$ is given by

$$p(\mathbf{y}|\mathcal{M}^{(b)}) = \sum_{i=1}^{K^{(b)}} \omega_i^{(b)} p(\mathbf{y}|z^{(b)} = i, \mathcal{M}^{(b)}), \tag{3}$$

where $z^{(b)} \sim \text{multinomial}(\omega_1^{(b)}, \cdots \omega_{K^{(b)}}^{(b)})$ is the hidden variable that indexes the mixture components. $p(\mathbf{y}|z = i, \mathcal{M}^{(b)})$ is the likelihood of $\mathbf{y}$ under the ith mixture component, as in (2), and $\omega_i^{(b)}$ is the mixture weight for the ith component. Likewise, the likelihood of the random sequence $\mathbf{y} \sim \mathcal{M}^{(r)}$ is

$$p(\mathbf{y}|\mathcal{M}^{(r)}) = \sum_{j=1}^{K^{(r)}} \omega_j^{(r)} p(\mathbf{y}|z^{(r)} = j, \mathcal{M}^{(r)}), \tag{4}$$

where $z^{(r)} \sim \text{multinomial}(\omega_1^{(r)}, \cdots, \omega_{K^{(r)}}^{(r)})$ is the hidden variable for indexing components in $\mathcal{M}^{(r)}$.

At a high level, the VHEM-H3M algorithm estimates the reduced H3M model $\mathcal{M}^{(r)}$ in (4) from *virtual* sequences distributed according to the base H3M model $\mathcal{M}^{(b)}$ in (3). From this estimation procedure, the VHEM algorithm provides:

1. a soft clustering of the original $K^{(b)}$ components into $K^{(r)}$ groups, where cluster membership is encoded in assignment variables that represent the *responsibility* of each reduced mixture component for each base mixture component, that is, $\hat{z}_{i,j} = p(z^{(r)} = j|z^{(b)} = i)$, for $i = 1, \ldots, K^{(b)}$ and $j = 1, \ldots, K^{(r)}$;

variables	*base model (b)*	*reduced model (r)*
index for HMM components	i	j
number of HMM components	$K^{(b)}$	$K^{(r)}$
HMM states	β	ρ
number of HMM states	S	S
HMM state sequence	$\boldsymbol{\beta} = \{\beta_1, \cdots, \beta_\tau\}$	$\boldsymbol{\rho} = \{\rho_1, \cdots, \rho_\tau\}$
index for component of GMM	m	ℓ
number of Gaussian components	M	M
models		
H3M	$\mathcal{M}^{(b)}$	$\mathcal{M}^{(r)}$
HMM component (of H3M)	$\mathcal{M}_i^{(b)}$	$\mathcal{M}_j^{(r)}$
GMM emission	$\mathcal{M}_{i,\beta}^{(b)}$	$\mathcal{M}_{j,\rho}^{(r)}$
Gaussian component (of GMM)	$\mathcal{M}_{i,\beta,m}^{(b)}$	$\mathcal{M}_{j,\rho,\ell}^{(r)}$
parameters		
H3M component weight	$\omega_i^{(b)}$	$\omega_j^{(r)}$
HMM initial state	$\pi^{(b),i}$	$\pi^{(r),j}$
HMM state transition matrix	$A^{(b),i}$	$A^{(r),j}$
GMM emission	$\{c_{\beta,m}^{(b),i}, \mu_{\beta,m}^{(b),i}, \Sigma_{\beta,m}^{(b),i}\}_{m=1}^M$	$\{c_{\rho,\ell}^{(r),j}, \mu_{\rho,\ell}^{(r),j}, \Sigma_{\rho,\ell}^{(r),j}\}_{\ell=1}^M$
probability distributions	*notation*	*short-hand*
HMM state sequence (b)	$p(\mathbf{x} = \boldsymbol{\beta} \vert z^{(b)} = i, \mathcal{M}^{(b)})$	$p(\boldsymbol{\beta} \vert \mathcal{M}_i^{(b)}) = \pi_{\boldsymbol{\beta}}^{(b),i}$
HMM state sequence (r)	$p(\mathbf{x} = \boldsymbol{\rho} \vert z^{(r)} = j, \mathcal{M}^{(r)})$	$p(\boldsymbol{\rho} \vert \mathcal{M}_j^{(r)}) = \pi_{\boldsymbol{\rho}}^{(r),j}$
HMM observation likelihood (r)	$p(\mathbf{y} \vert z^{(r)} = j, \mathcal{M}^{(r)})$	$p(\mathbf{y} \vert \mathcal{M}_j^{(r)})$
GMM emission likelihood (r)	$p(y_t \vert x_t = \rho, \mathcal{M}_j^{(r)})$	$p(y_t \vert \mathcal{M}_{j,\rho}^{(r)})$
Gaussian component likelihood (r)	$p(y_t \vert \zeta_t = \ell, x_t = \rho, \mathcal{M}_j^{(r)})$	$p(y_t \vert \mathcal{M}_{j,\rho,\ell}^{(r)})$
expectations		
HMM observation sequence (b)	$\mathrm{E}_{\mathbf{y} \vert z^{(b)}=i, \mathcal{M}^{(b)}}[\cdot]$	$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\cdot]$
GMM emission (b)	$\mathrm{E}_{y_t \vert x_t=\beta, \mathcal{M}_i^{(b)}}[\cdot]$	$\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\cdot]$
Gaussian component (b)	$\mathrm{E}_{y_t \vert \zeta_t=m, x_t=\beta, \mathcal{M}_i^{(b)}}[\cdot]$	$\mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\cdot]$
expected log-likelihood	*lower bound*	*variational distribution*
$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(Y_i \vert \mathcal{M}^{(r)})]$	$\mathcal{L}_{H3M}^i$	$q_i(z_i = j) = z_{ij}$
$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y} \vert \mathcal{M}_j^{(r)})]$	$\mathcal{L}_{HMM}^{i,j}$	$q^{i,j}(\boldsymbol{\rho} \vert \boldsymbol{\beta}) = \phi_{\boldsymbol{\rho} \vert \boldsymbol{\beta}}^{i,j}$ $= \phi_1^{i,j}(\rho_1 \vert \beta_1) \prod_{t=2}^{\tau} \phi_t^{i,j}(\rho_t \vert \rho_{t-1}, \beta_t)$
$\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y \vert \mathcal{M}_{j,\rho}^{(r)})]$	$\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$	$q_{\beta,\rho}^{i,j}(\zeta = \ell \vert m) = \eta_{\ell \vert m}^{(i,\beta),(j,\rho)}$

Table 1: Notation used in the derivation of the VHEM-H3M algorithm.

2. novel cluster centers represented by the individual mixture components of the reduced model in (4), that is, $p(\mathbf{y} \vert z^{(r)} = j, \mathcal{M}^{(r)})$ for $j = 1, \ldots, K^{(r)}$.

Finally, because we take the expectation over the virtual samples, the estimation is carried out in an efficient manner that requires only knowledge of the parameters of the base model, without the need of generating actual virtual samples.

3.1.1 Notation

We will always use i and j to index the components of the base model $\mathcal{M}^{(b)}$ and the reduced model $\mathcal{M}^{(r)}$, respectively. To reduce clutter, we will also use the short-hand notation $\mathcal{M}_i^{(b)}$ and $\mathcal{M}_j^{(r)}$ to denote the ith component of $\mathcal{M}^{(b)}$ and the jth component of $\mathcal{M}^{(r)}$, respectively. Hidden states of the HMMs are denoted with β for the base model $\mathcal{M}_i^{(b)}$, and with ρ for the reduced model $\mathcal{M}_j^{(r)}$.

The GMM emission models for each hidden state are denoted as $\mathcal{M}_{i,\beta}^{(b)}$ and $\mathcal{M}_{j,\rho}^{(r)}$. We will always use m and ℓ for indexing the individual Gaussian components of the GMM emissions of the base and reduced models, respectively. The individual Gaussian components are denoted as $\mathcal{M}_{i,\beta,m}^{(b)}$ for the base model, and $\mathcal{M}_{j,\rho,\ell}^{(r)}$ for the reduced model. Finally, we denote the parameters of ith HMM component of the base mixture model as $\mathcal{M}_i^{(b)} = \{\pi^{(b),i}, A^{(b),i}, \{\{c_{\beta,m}^{(b),i}, \mu_{\beta,m}^{(b),i}, \Sigma_{\beta,m}^{(b),i}\}_{m=1}^{M}\}_{\beta=1}^{S}\}$, and for the jth HMM in the reduced mixture as $\mathcal{M}_j^{(r)} = \{\pi^{(r),j}, A^{(r),j}, \{\{c_{\rho,\ell}^{(r),j}, \mu_{\rho,\ell}^{(r),j}, \Sigma_{\rho,\ell}^{(r),j}\}_{\ell=1}^{M}\}_{\rho=1}^{S}\}$.

When appearing in a probability distribution, the short-hand model notation (e.g., $\mathcal{M}_i^{(b)}$) always implies *conditioning* on the model being active. For example, we will use $p(\mathbf{y}|\mathcal{M}_i^{(b)})$ as short-hand for $p(\mathbf{y}|z^{(b)} = i, \mathcal{M}^{(b)})$, or $p(y_t|\mathcal{M}_{i,\beta}^{(b)})$ as short-hand for $p(y_t|x_t = \beta, z^{(b)} = i, \mathcal{M}^{(b)})$. Furthermore, we will use $\pi_{\boldsymbol{\beta}}^{(b),i}$ as short-hand for the probability of the state sequence $\boldsymbol{\beta}$ according to the base HMM component $\mathcal{M}_i^{(b)}$, that is, $p(\boldsymbol{\beta}|\mathcal{M}_i^{(b)})$, and likewise $\mathcal{M}_{\boldsymbol{\rho}}^{(r),j}$ for the reduced HMM component.

Expectations will also use the short-hand model notation to imply conditioning on the model. In addition, expectations are assumed to be taken with respect to the output variable ($\mathbf{y}$ or y_t), unless otherwise specified. For example, we will use $\mathrm{E}_{\mathcal{M}_i^{(b)}}[\cdot]$ as short-hand for $\mathrm{E}_{\mathbf{y}|z^{(b)}=i,\mathcal{M}^{(b)}}[\cdot]$.

Table 1 summarizes the notation used in the derivation, including the variable names, model parameters, and short-hand notations for probability distributions and expectations. The bottom of Table 1 also summarizes the variational lower bound and variational distributions, which will be introduced subsequently.

3.2 Variational HEM Algorithm

To learn the reduced model in (4), we consider a set of N *virtual* samples, distributed according to the base model $\mathcal{M}^{(b)}$ in (3), such that $N_i = N\omega_i^{(b)}$ samples are drawn from the ith component. We denote the set of N_i virtual samples for the ith component as $Y_i = \{\mathbf{y}^{(i,m)}\}_{m=1}^{N_i}$, where $\mathbf{y}^{(i,m)} \sim \mathcal{M}_i^{(b)}$, and the entire set of N samples as $Y = \{Y_i\}_{i=1}^{K^{(b)}}$. Note that, in this formulation, we are not considering virtual samples $\{\mathbf{x}^{(i,m)}, \mathbf{y}^{(i,m)}\}$ for each base component, according to its joint distribution $p(\mathbf{x}, \mathbf{y}|\mathcal{M}_i^{(b)})$. The reason is that the hidden-state space of each base mixture component $\mathcal{M}_i^{(b)}$ may have a different representation (e.g., the numbering of the hidden states may be permuted between the components). This mismatch will cause problems when the parameters of $\mathcal{M}_j^{(r)}$ are computed from virtual samples of the hidden states of $\{\mathcal{M}_i^{(b)}\}_{i=1}^{K^{(b)}}$. Instead, we treat $X_i = \{\mathbf{x}^{(i,m)}\}_{m=1}^{N_i}$ as "missing"

information, and estimate them in the E-step. The log-likelihood of the virtual samples is

$$\log p(Y|\mathcal{M}^{(r)}) = \sum_{i=1}^{K^{(b)}} \log p(Y_i|\mathcal{M}^{(r)}), \tag{5}$$

where, in order to obtain a consistent clustering, we assume the entirety of samples Y_i is assigned to the same component of the reduced model (Vasconcelos and Lippman, 1998).

The original formulation of HEM (Vasconcelos and Lippman, 1998) maximizes (5) with respect to $\mathcal{M}^{(r)}$, and uses the law of large numbers to turn the virtual samples Y_i into an expectation over the base model components $\mathcal{M}_i^{(b)}$. In this paper, we will start with a different objective function to derive the VHEM algorithm. To estimate $\mathcal{M}^{(r)}$, we will maximize the average log-likelihood of all possible virtual samples, weighted by their likelihood of being generated by $\mathcal{M}_i^{(b)}$, that is, the *expected* log-likelihood of the virtual samples,

$$\mathcal{J}(\mathcal{M}^{(r)}) = \mathrm{E}_{\mathcal{M}^{(b)}}\left[\log p(Y|\mathcal{M}^{(r)})\right] = \sum_{i=1}^{K^{(b)}} \mathrm{E}_{\mathcal{M}_i^{(b)}}\left[\log p(Y_i|\mathcal{M}^{(r)})\right], \tag{6}$$

where the expectation is over the base model components $\mathcal{M}_i^{(b)}$. Maximizing (6) will eventually lead to the same estimate as maximizing (5), but allows us to strictly preserve the variational lower bound, which would otherwise be ruined when applying the law of large numbers to (5).

A general approach to deal with maximum likelihood estimation in the presence of hidden variables (which is the case for H3Ms) is the EM algorithm (Dempster et al., 1977). In the traditional formulation the EM algorithm is presented as an alternation between an expectation step (E-step) and a maximization step (M-step). In this work, we take a variational perspective (Neal and Hinton, 1998; Wainwright and Jordan, 2008; Csiszár and Tusnády, 1984), which views each step as a maximization step. The variational E-step first obtains a family of lower bounds to the (expected) log-likelihood (i.e., to Equation 6), indexed by variational parameters, and then optimizes over the variational parameters to find the tightest bound. The corresponding M-step then maximizes the lower bound (with the variational parameters fixed) with respect to the model parameters. One advantage of the variational formulation is that it readily allows for useful extensions to the EM algorithm, such as replacing a difficult inference in the E-step with a variational approximation. In practice, this is achieved by restricting the maximization in the variational E-step to a smaller domain for which the lower bound is tractable.

The EM algorithm with variational E-step is guaranteed to converge (Gunawardana and Byrne, 2005). Despite the approximation prevents convergence to local maxima of the data log-likelihood (Gunawardana and Byrne, 2005), the algorithm still performs well empirically, as shown in Section 5 and Section 6.

3.2.1 Lower Bound to an Expected Log-likelihood

Before proceeding with the derivation of VHEM for H3Ms, we first need to derive a lower-bound to an expected log-likelihood term, for example, (6). Our derivation starts from a variational lower-bound to a log-likelihood (as opposed to an *expected* log-likelihood), a

standard tool in machine learning (Jordan et al., 1999; Jaakkola, 2000), which we briefly review next. In all generality, let $\{O, H\}$ be the observation and hidden variables of a probabilistic model, respectively, where $p(H)$ is the distribution of the hidden variables, $p(O|H)$ is the conditional likelihood of the observations, and $p(O) = \sum_H p(O|H)p(H)$ is the observation likelihood. We can define a *variational lower bound* to the observation log-likelihood (Jordan et al., 1999; Jaakkola, 2000):

$$\begin{aligned}\log p(O) &\geq \log p(O) - D(q(H)||p(H|O)) \\ &= \sum_H q(H) \log \frac{p(H)p(O|H)}{q(H)},\end{aligned}$$

where $p(H|O)$ is the posterior distribution of H given observation O, and $D(p\|q) = \int p(y) \log \frac{p(y)}{q(y)} dy$ is the Kullback-Leibler (KL) divergence between two distributions, p and q. We introduce a variational distribution $q(H)$, which approximates the posterior distribution, where $\sum_H q(H) = 1$ and $q(H) \geq 0$. When the variational distribution equals the true posterior, $q(H) = P(H|O)$, then the KL divergence is zero, and hence the lower-bound reaches $\log p(O)$. When the true posterior cannot be computed, then typically q is restricted to some set of approximate posterior distributions $\mathcal{Q}$ that are tractable, and the best lower-bound is obtained by maximizing over $q \in \mathcal{Q}$,

$$\log p(O) \geq \max_{q \in \mathcal{Q}} \sum_H q(H) \log \frac{p(H)p(O|H)}{q(H)}. \tag{7}$$

From the standard lower bound in (7), we can now derive a lower bound to an expected log-likelihood expression. Let $\mathrm{E}_b[\cdot]$ be the expectation with respect to O with some distribution $p_b(O)$. Since $p_b(O)$ is non-negative, taking the expectation on both sides of (7) yields,

$$\mathrm{E}_b\left[\log p(O)\right] \geq \mathrm{E}_b\left[\max_{q \in Q} \sum_H q(H) \log \frac{p(H)p(O|H)}{q(H)}\right] \tag{8}$$

$$\geq \max_{q \in Q} \mathrm{E}_b\left[\sum_H q(H) \log \frac{p(H)p(O|H)}{q(H)}\right] \tag{9}$$

$$= \max_{q \in Q} \sum_H q(H) \left\{\log \frac{p(H)}{q(H)} + \mathrm{E}_b\left[\log p(O|H)\right]\right\}, \tag{10}$$

where (9) follows from Jensen's inequality (i.e., $f(\mathrm{E}[x]) \leq \mathrm{E}[f(x)]$ when f is convex), and the convexity of the max function. Hence, (10) is a variational lower bound on the expected log-likelihood, which depends on the family of variational distributions Q.

In (8) we are computing the best lower-bound (7) to $\log p(O)$ *individually* for each value of the observation variable O, which in general corresponds to different optimal $q^* \in \mathcal{Q}$ for different values of O. Note that the expectation in (8) is not analytically tractable when $p(O)$ is a mixture model (i.e., it is the expected log-likelihood of a mixture). Hence, we treat the ensemble of observations $O \sim p_b$ as a whole, and in (9) find a single $q^* \in \mathcal{Q}$ for which the lower bound is best on average. Mathematically, this correspond to using Jensen inequality to pass from (8) to (9), which shows that the additional approximation makes the lower-bound looser.

3.2.2 Variational Lower Bound

We now derive a lower bound to the expected log-likelihood cost function in (6). The derivation will proceed by successively applying the lower bound from (10) to each expected log-likelihood term that arises. This will result in a set of nested lower bounds.

A variational lower bound to the expected log-likelihood of the virtual samples in (6) is obtained by lower bounding each of the expectation terms $\mathrm{E}_{\mathcal{M}_i^{(b)}}$ in the sum,

$$\mathcal{J}(\mathcal{M}^{(r)}) = \sum_{i=1}^{K^{(b)}} \mathrm{E}_{\mathcal{M}_i^{(b)}} \left[\log p(Y_i|\mathcal{M}^{(r)})\right] \geq \sum_{i=1}^{K^{(b)}} \mathcal{L}^i_{H3M}, \tag{11}$$

where we define three nested lower bounds, corresponding to different model elements (the H3M, the component HMMs, and the emission GMMs):

$$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(Y_i|\mathcal{M}^{(r)})] \geq \mathcal{L}^i_{H3M}, \tag{12}$$

$$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})] \geq \mathcal{L}^{i,j}_{HMM}, \tag{13}$$

$$\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})] \geq \mathcal{L}^{(i,\beta),(j,\rho)}_{GMM}. \tag{14}$$

In (12), the first lower bound, $\mathcal{L}^i_{H3M}$, is on the expected log-likelihood of an H3M $\mathcal{M}^{(r)}$ with respect to an HMM $\mathcal{M}_i^{(b)}$. Because $p(Y_i|\mathcal{M}^{(r)})$ is the likelihood under a mixture of HMMs, as in (4), where the observation variable is Y_i and the hidden variable is z_i (the assignment of Y_i to a component of $\mathcal{M}^{(r)}$), its expectation cannot be calculated directly. Hence, we introduce the variational distribution $q_i(z_i)$ and apply (10) to (12), yielding the lower bound (see Appendix A),

$$\mathcal{L}^i_{H3M} = \max_{q_i} \sum_j q_i(z_i = j) \left\{ \log \frac{p(z_i = j|\mathcal{M}^{(r)})}{q_i(z_i = j)} + N_i \mathcal{L}^{i,j}_{HMM} \right\}. \tag{15}$$

The lower bound in (15) depends on the second lower bound (Eq. 13), $\mathcal{L}^{i,j}_{HMM}$, which is on the expected log-likelihood of an HMM $\mathcal{M}_j^{(r)}$, averaged over observation sequences from a *different* HMM $\mathcal{M}_i^{(b)}$. Although the data log-likelihood $\log p(\mathbf{y}|\mathcal{M}_j^{(r)})$ can be computed exactly using the forward algorithm (Rabiner and Juang, 1993), calculating its expectation is not analytically tractable since an observation sequence $\mathbf{y}$ from a HMM $\mathcal{M}_j^{(r)}$ is essentially an observation from a mixture model.[1]

To calculate the lower bound $\mathcal{L}^{i,j}_{HMM}$ in (13), we first rewrite the expectation $\mathrm{E}_{\mathcal{M}_i^{(b)}}$ in (13) to explicitly marginalize over the state sequence $\boldsymbol{\beta}$ of $\mathcal{M}_i^{(b)}$, and then apply (10) where the hidden variable is the state sequence $\boldsymbol{\rho}$ of $\mathcal{M}_j^{(r)}$, yielding (see Appendix A)

$$\mathcal{L}^{i,j}_{HMM} = \sum_{\boldsymbol{\beta}} \pi^{(b),i}_{\boldsymbol{\beta}} \max_{q^{i,j}} \sum_{\boldsymbol{\rho}} q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{p(\boldsymbol{\rho}|\mathcal{M}_j^{(r)})}{q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_t \mathcal{L}^{(i,\beta_t),(j,\rho_t)}_{GMM} \right\}, \tag{16}$$

1. For an observation sequence of length τ, an HMM with S states can be considered as a mixture model with $O(S^\tau)$ components.

where we introduce a variational distribution $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$ on the state sequence $\boldsymbol{\rho}$, which depends on a particular sequence $\boldsymbol{\beta}$ from $\mathcal{M}_i^{(b)}$. As before, (16) depends on another nested lower bound, $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ in (14), which is on the expected log-likelihood of a GMM emission density $\mathcal{M}_{j,\rho}^{(r)}$ with respect to another GMM $\mathcal{M}_{i,\beta}^{(b)}$. This lower bound does not depend on time, as we have assumed that the emission densities are time-invariant.

Finally, we obtain the lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ for (14), by explicitly marginalizing over the GMM hidden assignment variable in $\mathcal{M}_{i,\beta}^{(b)}$ and then applying (10) to the expectation of the GMM emission distribution $p(y|\mathcal{M}_{j,\rho}^{(r)})$, yielding (see Appendix A),

$$\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)} = \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \max_{q_{\beta,\rho}^{i,j}} \sum_{\zeta=1}^{M} q_{\beta,\rho}^{i,j}(\zeta|m) \left\{ \log \frac{p(\zeta|\mathcal{M}_{j,\rho}^{(r)})}{q_{\beta,\rho}^{i,j}(\zeta|m)} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\zeta}^{(r)})] \right\}, \quad (17)$$

where we introduce the variational distribution $q_{\beta,\rho}^{i,j}(\zeta|m)$, which is conditioned on the observation y arising from the mth component in $\mathcal{M}_{i,\beta}^{(b)}$. In (17), the term $\mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})]$ is the expected log-likelihood of the Gaussian distribution $\mathcal{M}_{j,\rho,\ell}^{(r)}$ with respect to the Gaussian $\mathcal{M}_{i,\beta,m}^{(b)}$, which has a closed-form solution (see Section 3.3.1).

In summary, we have derived a variational lower bound to the expected log-likelihood of the virtual samples, which is given by (11). This lower bound is composed of three nested lower bounds in (15), (16), and (17), corresponding to different model elements (the H3M, the component HMMs, and the emission GMMs), where $q_i(z_i)$, $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$, and $q_{\beta,\rho}^{i,j}(\zeta|m)$ are the corresponding variational distributions. Finally, the variational HEM algorithm for HMMs consists of two alternating steps:

- (variational E-step) given $\mathcal{M}^{(r)}$, calculate the variational distributions $q_{\beta,\rho}^{i,j}(\zeta|m)$, $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$, and $q_i(z_i)$ for the lower bounds in (17), (16), and (15);
- (M-step) update the model parameters via $\mathcal{M}^{(r)*} = \mathrm{argmax}_{\mathcal{M}^{(r)}} \sum_{i=1}^{K^{(b)}} \mathcal{L}_{H3M}^{i}$.

In the following subsections, we derive the E- and M-steps of the algorithm. The entire procedure is summarized in Algorithm 1.

3.3 Variational E-Step

The variational E-step consists of finding the variational distributions that maximize the lower bounds in (17), (16), and (15). In particular, given the nesting of the lower bounds, we proceed by first maximizing the GMM lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ for each pair of emission GMMs in the base and reduced models. Next, the HMM lower bound $\mathcal{L}_{HMM}^{i,j}$ is maximized for each pair of HMMs in the base and reduced models, followed by maximizing the H3M lower bound $\mathcal{L}_{H3M}^{i}$ for each base HMM. Finally, a set of summary statistics are calculated, which will be used in the M-step.

3.3.1 Variational Distributions

We first consider the forms of the three variational distributions, as well as the optimal parameters to maximize the corresponding lower bounds.

Algorithm 1 VHEM algorithm for H3Ms

Input: base H3M $\mathcal{M}^{(b)} = \{\omega_i^{(b)}, \mathcal{M}_i^{(b)}\}_{i=1}^{K^{(b)}}$, number of virtual samples N.
Initialize reduced H3M $\mathcal{M}^{(r)} = \{\omega_j^{(r)}, \mathcal{M}_j^{(r)}\}_{j=1}^{K^{(r)}}$.
repeat
  {Variational E-step}
  Compute optimal variational distributions and variational lower bounds:
  for each pair of HMMs $\mathcal{M}_i^{(s)}$ and $\mathcal{M}_j^{(r)}$
    for each pair of emission GMMs for state β of $\mathcal{M}_i^{(s)}$ and ρ of $\mathcal{M}_j^{(r)}$:
      Compute optimal variational distributions $\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}$ as in (18)
      Compute optimal lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ to expected log-likelihood as in (22)
    Compute optimal variational distributions for HMMs as in Appendin B
      $\hat{\phi}_1^{i,j}(\rho_1|\beta_1), \quad \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1}, \beta_t)$ for $t = \tau, \ldots, 2$
    Compute optimal lower bound $\mathcal{L}_{HMM}^{i,j}$ to expected log-likelihood as in (21)
  Compute optimal assignment probabilities:

$$\hat{z}_{ij} = \frac{\omega_j^{(r)} \exp(N\omega_i^{(b)} \mathcal{L}_{HMM}^{i,j})}{\sum_{j'} \omega_{j'}^{(r)} \exp(N\omega_i^{(b)} \mathcal{L}_{HMM}^{i,j'})}$$

  Compute aggregate summary statistics for each pair of HMMs $\mathcal{M}_i^{(s)}$ and $\mathcal{M}_j^{(r)}$ as in Section 3.3.3:

$$\hat{\nu}_1^{i,j}(\rho) = \sum_{\beta=1}^{S} \nu_1^{i,j}(\rho, \beta), \quad \hat{\nu}^{i,j}(\rho, \beta) = \sum_{t=1}^{\tau} \nu_t^{i,j}(\rho, \beta), \quad \hat{\xi}^{i,j}(\rho, \rho') = \sum_{t=2}^{\tau} \sum_{\beta=1}^{S} \xi_t^{i,j}(\rho, \rho', \beta)$$

  {M-step}
  For each component $\mathcal{M}_j^{(r)}$, recompute parameters using (24)-(28).
until convergence
Output: reduced H3M $\{\omega_j^{(r)}, \mathcal{M}_j^{(a)}\}_{j=1}^{K^{(r)}}$.

GMM: For the GMM lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$, we assume each variational distribution has the form (Hershey et al., 2007)

$$q_{\beta,\rho}^{i,j}(\zeta = l|m) = \eta_{\ell|m}^{(i,\beta),(j,\rho)},$$

where $\sum_{\ell=1}^{M} \eta_{\ell|m}^{(i,\beta),(j,\rho)} = 1$, and $\eta_{\ell|m}^{(i,\beta),(j,\rho)} \geq 0$, $\forall \ell$. Intuitively, $\boldsymbol{\eta}^{(i,\beta),(j,\rho)}$ is the responsibility matrix between each pair of Gaussian components in the GMMs $\mathcal{M}_{i,\beta}^{(b)}$ and $\mathcal{M}_{j,\rho}^{(r)}$, where $\eta_{\ell|m}^{(i,\beta),(j,\rho)}$ represents the probability that an observation from component m of $\mathcal{M}_{i,\beta}^{(b)}$ corresponds to component ℓ of $\mathcal{M}_{j,\rho}^{(r)}$. Substituting into (17) and maximizing the variational

parameters yields (see Appendix B)

$$\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)} = \frac{c_{\rho,\ell}^{(r),j} \exp\left\{ \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right\}}{\sum_{\ell'} c_{\rho,\ell'}^{(r),j} \exp\left\{ \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\ell'}^{(r)})] \right\}}, \tag{18}$$

where the expected log-likelihood of a Gaussian $\mathcal{M}_{j,\rho,\ell}^{(r)}$ with respect to another Gaussian $\mathcal{M}_{i,\beta,m}^{(b)}$ is computable in closed-form (Penny and Roberts, 2000),

$$\begin{aligned}\mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] = &-\frac{d}{2}\log 2\pi - \frac{1}{2}\log\left|\Sigma_{\rho,\ell}^{(r),j}\right| - \frac{1}{2}\mathrm{tr}\left((\Sigma_{\rho,\ell}^{(r),j})^{-1}\Sigma_{\beta,m}^{(b),i}\right) \\ &- \frac{1}{2}(\mu_{\rho,\ell}^{(r),j} - \mu_{\beta,m}^{(b),i})^T(\Sigma_{\rho,\ell}^{(r),j})^{-1}(\mu_{\rho,\ell}^{(r),j} - \mu_{\beta,m}^{(b),i}).\end{aligned}$$

HMM: For the HMM lower bound $\mathcal{L}_{HMM}^{i,j}$, we assume each variational distribution takes the form of a Markov chain,

$$q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) = \phi^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) = \phi_1^{i,j}(\rho_1|\beta_1)\prod_{t=2}^{\tau}\phi_t^{i,j}(\rho_t|\rho_{t-1},\beta_t),$$

where $\sum_{\rho_1=1}^{S}\phi_1^{i,j}(\rho_1|\beta_1) = 1$, and $\sum_{\rho_t=1}^{S}\phi_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) = 1$, and all the factors are non-negative. The variational distribution $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$ represents the probability of the state sequence $\boldsymbol{\rho}$ in HMM $\mathcal{M}_j^{(r)}$, when $\mathcal{M}_j^{(r)}$ is used to explain the *observation* sequence generated by $\mathcal{M}_i^{(b)}$ that evolved through state sequence $\boldsymbol{\beta}$.

Substituting $\phi^{i,j}$ into (16), the maximization with respect to $\phi_t^{i,j}(\rho_t|\rho_{t-1},\beta_t)$ and $\phi_1^{i,j}(\rho_1|\beta_1)$ is carried out independently for each pair (i,j), and follows (Hershey et al., 2007). This is further detailed in Appendix B. By separating terms and breaking up the summation over $\boldsymbol{\beta}$ and $\boldsymbol{\rho}$, the optimal $\hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t)$ and $\hat{\phi}_1^{i,j}(\rho_1|\beta_1)$ can be obtained using an efficient recursive iteration (similar to the forward algorithm).

H3M: For the H3M lower bound $\mathcal{L}_{H3M}^{i}$, we assume variational distributions of the form $q_i(z_i = j) = z_{ij}$, where $\sum_{j=1}^{K^{(r)}} z_{ij} = 1$, and $z_{ij} \geq 0$. Substituting z_{ij} into (15), and maximizing variational parameters are obtained as (see Appendix B)

$$\hat{z}_{ij} = \frac{\omega_j^{(r)} \exp(N_i \mathcal{L}_{HMM}^{i,j})}{\sum_{j'} \omega_{j'}^{(r)} \exp(N_i \mathcal{L}_{HMM}^{i,j'})}. \tag{19}$$

Note that in the standard HEM algorithm (Vasconcelos and Lippman, 1998; Chan et al., 2010a), the assignment probabilities z_{ij} are based on the expected log-likelihoods of the components, (e.g., $\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]$ for H3Ms). For the variational HEM algorithm, these expectations are now replaced with their lower bounds (in our case, $\mathcal{L}_{HMM}^{i,j}$).

3.3.2 Lower Bound

Substituting the optimal variational distributions into (15), (16), and (17) gives the lower bounds,

$$\mathcal{L}^{i}_{H3M} = \sum_{j} \hat{z}_{ij} \left\{ \log \frac{\omega^{(r)}_{j}}{\hat{z}_{ij}} + N_i \mathcal{L}^{i,j}_{HMM} \right\}, \tag{20}$$

$$\mathcal{L}^{i,j}_{HMM} = \sum_{\boldsymbol{\beta}} \pi^{(b),i}_{\boldsymbol{\beta}} \sum_{\boldsymbol{\rho}} \hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{\pi^{(r),j}_{\boldsymbol{\rho}}}{\hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_{t} \mathcal{L}^{(i,\beta_t),(j,\rho_t)}_{GMM} \right\}, \tag{21}$$

$$\mathcal{L}^{(i,\beta),(j,\rho)}_{GMM} = \sum_{m=1}^{M} c^{(b),i}_{\beta,m} \sum_{\ell=1}^{M} \hat{\eta}^{(i,\beta),(j,\rho)}_{\ell|m} \left\{ \log \frac{c^{(r),j}_{\rho,\ell}}{\hat{\eta}^{(i,\beta),(j,\rho)}_{\ell|m}} + \mathrm{E}_{\mathcal{M}^{(b)}_{i,\beta,m}} [\log p(y|\mathcal{M}^{(r)}_{j,\rho,\ell})] \right\}. \tag{22}$$

The lower bound $\mathcal{L}^{i,j}_{HMM}$ requires summing over all sequences $\boldsymbol{\beta}$ and $\boldsymbol{\rho}$. This summation can be computed efficiently along with $\hat{\phi}^{i,j}_t(\rho_t|\rho_{t-1},\beta_t)$ and $\hat{\phi}^{i,j}_1(\rho_1|\beta_1)$ using a recursive algorithm from Hershey et al. (2007). This is described in Appendix B.

3.3.3 Summary Statistics

After calculating the optimal variational distributions, we calculate the following summary statistics, which are necessary for the M-step:

$$\nu^{i,j}_1(\rho_1,\beta_1) = \pi^{(b),i}_{\beta_1} \hat{\phi}^{i,j}_1(\rho_1|\beta_1),$$

$$\xi^{i,j}_t(\rho_{t-1},\rho_t,\beta_t) = \left(\sum_{\beta_{t-1}=1}^{S} \nu^{i,j}_{t-1}(\rho_{t-1},\beta_{t-1})\, a^{(b),i}_{\beta_{t-1},\beta_t} \right) \hat{\phi}^{i,j}_t(\rho_t|\rho_{t-1},\beta_t), \text{ for } t = 2,\ldots,\tau,$$

$$\nu^{i,j}_t(\rho_t,\beta_t) = \sum_{\rho_{t-1}=1}^{S} \xi^{i,j}_t(\rho_{t-1},\rho_t,\beta_t), \text{ for } t = 2,\ldots,\tau,$$

and the aggregate statistics

$$\hat{\nu}^{i,j}_1(\rho) = \sum_{\beta=1}^{S} \nu^{i,j}_1(\rho,\beta), \tag{23}$$

$$\hat{\nu}^{i,j}(\rho,\beta) = \sum_{t=1}^{\tau} \nu^{i,j}_t(\rho,\beta),$$

$$\hat{\xi}^{i,j}(\rho,\rho') = \sum_{t=2}^{\tau} \sum_{\beta=1}^{S} \xi^{i,j}_t(\rho,\rho',\beta).$$

The statistic $\hat{\nu}^{i,j}_1(\rho)$ is the expected number of times that the HMM $\mathcal{M}^{(r)}_j$ starts from state ρ, when modeling sequences generated by $\mathcal{M}^{(b)}_i$. The quantity $\hat{\nu}^{i,j}(\rho,\beta)$ is the expected number of times that the HMM $\mathcal{M}^{(r)}_j$ is in state ρ when the HMM $\mathcal{M}^{(b)}_i$ is in state β, when both HMMs are modeling sequences generated by $\mathcal{M}^{(b)}_i$. Similarly, the quantity $\hat{\xi}^{i,j}(\rho,\rho')$

is the expected number of transitions from state ρ to state ρ' of the HMM $\mathcal{M}_j^{(r)}$, when modeling sequences generated by $\mathcal{M}_i^{(b)}$.

3.4 M-Step

In the M-step, the lower bound in (11) is maximized with respect to the parameters $\mathcal{M}^{(r)}$,

$$\mathcal{M}^{(r)^*} = \operatorname*{argmax}_{\mathcal{M}^{(r)}} \sum_{i=1}^{K^{(b)}} \mathcal{L}_{H3M}^{i}.$$

The derivation of the maximization is presented in Appendix C. Each mixture component of $\mathcal{M}^{(r)}$ is updated independently according to

$$\omega_j^{(r)^*} = \frac{\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}}{K^{(b)}}, \tag{24}$$

$$\pi_\rho^{(r),j^*} = \frac{\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)}\hat{\nu}_1^{i,j}(\rho)}{\sum_{\rho'=1}^{S}\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)}\hat{\nu}_1^{i,j}(\rho'))}, \quad a_{\rho,\rho'}^{(r),j^*} = \frac{\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)}\hat{\xi}^{i,j}(\rho,\rho')}{\sum_{\sigma=1}^{S}\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)}\hat{\xi}^{i,j}(\rho,\sigma)}, \tag{25}$$

$$c_{\rho,\ell}^{(r),j^*} = \frac{\Omega_{j,\rho}\left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}\right)}{\sum_{\ell'=1}^{M}\Omega_{j,\rho}\left(\hat{\eta}_{\ell'|m}^{(i,\beta),(j,\rho)}\right)}, \qquad \mu_{\rho,\ell}^{(r),j^*} = \frac{\Omega_{j,\rho}\left(\eta_{\ell|m}^{(i,\beta),(j,\rho)}\ \mu_{\beta,m}^{(b),i}\right)}{\Omega_{j,\rho}\left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}\right)}, \tag{26}$$

$$\Sigma_{\rho,\ell}^{(r),j^*} = \frac{\Omega_{j,\rho}\left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}\left[\Sigma_{\beta,m}^{(b),i} + (\mu_{\beta,m}^{(b),i} - \mu_{\rho,\ell}^{(r),j})(\mu_{\beta,m}^{(b),i} - \mu_{\rho,\ell}^{(r),j})^T\right]\right)}{\Omega_{j,\rho}\left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}\right)}, \tag{27}$$

where $\Omega_{j,\rho}(\cdot)$ is the weighted sum operator over all base models, HMM states, and GMM components (i.e., over all tuples (i,β,m)),

$$\Omega_{j,\rho}(f(i,\beta,m)) = \sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)} \sum_{\beta=1}^{S} \hat{\nu}^{i,j}(\rho,\beta) \sum_{m=1}^{M} c_{\beta,m}^{(b),i}\, f(i,\beta,m). \tag{28}$$

The terms $\pi_\rho^{(r),j}$ and $a_{\rho,\rho'}^{(r),j}$ are elements of the initial state prior and transition matrix, $\pi^{(r),j}$ and $A^{(r),j}$. Note that the covariance matrices of the reduced models in (27) include an additional outer-product term, which acts to regularize the covariances of the base models. This regularization effect derives from the E-step, which averages all possible observations from the base model.

4. Applications and Related Work

In the previous section, we derived the VHEM-H3M algorithm to cluster HMMs. We now discuss various applications of the algorithm (Section 4.1), and then present some literature that is related to HMM clustering (Section 4.2).

4.1 Applications of the VHEM-H3M Algorithm

The proposed VHEM-H3M algorithm clusters HMMs *directly* through the distributions they represent, and learns *novel* HMM cluster centers that compactly represent the structure of each cluster.

An application of the VHEM-H3M algorithm is in *hierarchical clustering* of HMMs. In particular, the VHEM-H3M algorithm is used recursively on the HMM cluster centers, to produce a bottom-up hierarchy of the input HMMs. Since the cluster centers condense the structure of the clusters they represent, the VHEM-H3M algorithm can implicitly leverage rich information on the underlying structure of the clusters, which is expected to impact positively the quality of the resulting hierarchical clustering.

Another application of VHEM is for efficient estimation of H3Ms from data, by using a *hierarchical estimation procedure* to break the learning problem into smaller pieces. First, a data set is split into small (non-overlapping) portions and intermediate HMMs are learned for each portion, via standard EM. Then, the final model is estimated from the intermediate models using the VHEM-H3M algorithm. Because VHEM and standard EM are based on similar maximum-likelihood principles, it drives model estimation towards similar optimal parameter values as performing EM estimation directly on the full data set. However, compared to direct EM estimation, VHEM-H3M is more memory- and time-efficient. First, it no longer requires storing in memory the entire data set during parameter estimation. Second, it does not need to evaluate the likelihood of all the samples at each iteration, and converges to effective estimates in shorter times. Note that even if a parallel implementation of EM could effectively handle the high memory requirements, a parallel-VHEM will still require fewer resources than a parallel-EM.

In addition, for the hierarchical procedure, the estimation of the intermediate models can be easily parallelized, since they are learned independently of each other. Finally, hierarchical estimation allows for efficient model updating when adding new data. Assuming that the previous intermediate models have been saved, re-estimating the H3M requires learning the intermediate models of only the new data, followed by running VHEM again. Since estimation of the intermediate models is typically as computationally intensive as the VHEM stage, reusing the previous intermediate models will lead to considerable computational savings when re-estimating the H3M.

In hierarchical estimation (EM on each time-series, VHEM on intermediate models), VHEM implicitly averages over all possible observations (virtual variations of each time-series) compatible with the intermediate models. We expect this to regularize estimation, which may result in models that generalize better (compared to estimating models with direct EM). Lastly, the "virtual" samples (i.e., sequences), which VHEM implicitly generates for maximum-likelihood estimation, need not be of the same length as the actual input data for estimating the intermediate models. Making the virtual sequences relatively short will positively impact the run time of each VHEM iteration. This may be achieved without loss of modeling accuracy, as show in Section 6.3.

4.2 Related Work

Jebara et al. (2007)'s approach to clustering HMMs consists of applying spectral clustering to a probability product kernel (PPK) matrix between HMMs—we will refer to it as PPK-

SC. In particular, the PPK similarity between two HMMs, $\mathcal{M}^{(a)}$ and $\mathcal{M}^{(b)}$, is defined as

$$k(a,b) = \int p(\mathbf{y}|\mathcal{M}^{(a)})^{\lambda}\, p(\mathbf{y}|\mathcal{M}^{(b)})^{\lambda} d\mathbf{y}, \tag{29}$$

where λ is a scalar, and τ is the length of "virtual" sequences. The case $\lambda = \frac{1}{2}$ corresponds to the Bhattacharyya affinity. While this approach indirectly leverages the probability distributions represented by the HMMs (i.e., the PPK affinity is computed from the probability distributions of the HMMs) and has proven successful in grouping HMMs into similar clusters (Jebara et al., 2007), it has several limitations. First, the spectral clustering algorithm cannot produce novel HMM cluster centers to represent the clusters, which is suboptimal for several applications of HMM clustering. For example, when implementing hierarchical clustering in the spectral embedding space (e.g., using hierarchical k-means clustering), clusters are represented by *single* points in the embedding space. This may fail to capture information on the local structure of the clusters that, when using VHEM-H3M, would be encoded by the novel HMM cluster centers. Hence, we expect VHEM-H3M to produce better hierarchical clustering than the spectral clustering algorithm, especially at higher levels of the hierarchy. This is because, when building a new level, VHEM can leverage more information from the lower levels, as encoded in the HMM cluster centers.

One simple extension of PPK-SC to obtain a HMM cluster center is to select the input HMM that the spectral clustering algorithm maps closest to the spectral clustering center. However, with this method, the HMM cluster centers are limited to be one of the *existing* input HMMs (i.e., similar to the k-medoids algorithm by Kaufman and Rousseeuw 1987), instead of the HMMs that optimally condense the structure of the clusters. Therefore, we expect the *novel* HMM cluster centers learned by VHEM-H3M to better represent the clusters. A more involved, "hybrid" solution is to learn the HMM cluster centers with VHEM-H3M *after* obtaining clusters with PPK-SC—using the VHEM-H3M algorithm to summarize all the HMMs within each PPK-SC cluster into a single HMM. However, we expect our VHEM-H3M algorithm to learn more accurate clustering models, since it jointly learns the clustering and the HMM centers, by optimizing a single objective function (i.e., the lower bound to the expected log-likelihood in Equation 11).

A second drawback of the spectral clustering algorithm is that the construction and the inversion of the similarity matrix between the input HMMs is a costly operation when their number is large[2] (e.g., see the experiment on H3M density estimation on the music data in Section 6.1). Therefore, we expect VHEM-H3M to be computationally more efficient than the spectral clustering algorithm since, by *directly* operating on the probability distributions of the HMMs, it does not require the construction of an initial embedding or any costly matrix operation on large kernel matrices.

Finally, as Jebara et al. (2004) note, the exact computation of (29) cannot be carried out efficiently, unless $\lambda = 1$. For different values of λ,[3] Jebara et al. (2004) propose to *approximate* $k(a,b)$ with an alternative kernel function that can be efficiently computed;

2. The computational complexity of large spectral clustering problems can be alleviated by means of numerical techniques for the solutions of eigenfunction problems such as the Nyström method (Nyström, 1930; Fowlkes et al., 2004), or by sampling only part of the similarity matrix and using a sparse eigensonver (Achlioptas et al., 2002).
3. The experimental results in Jebara et al. (2004) and Jebara et al. (2007) suggest to use $\lambda < 1$.

this alternative kernel function, however, is not guaranteed to be invariant to different but equivalent representations of the hidden state process (Jebara et al., 2004).[4] Alternative approximations for the Bhattacharyya setting (i.e., $\lambda = \frac{1}{2}$) have been proposed by Hershey and Olsen (2008).

Note that spectral clustering algorithms (similar to the one by Jebara et al. 2007) can be applied to kernel (similarity) matrices that are based on other affinity scores between HMM distributions than the PPK similarity of Jebara et al. (2004). Examples can be found in earlier work on HMM-based clustering of time-series, such as by Juang and Rabiner (1985), Lyngso et al. (1999), Bahlmann and Burkhardt (2001), Panuccio et al. (2002). In particular, Juang and Rabiner (1985) propose to approximate the (symmetrised) log-likelihood between two HMM distributions by computing the log-likelihood of real samples generated by one model under the other.[5] Extensions of the work of Juang and Rabiner (1985) have been proposed by Zhong and Ghosh (2003) and Yin and Yang (2005). In this work we do not pursue a comparison of the various similarity functions, but implement spectral clustering only based on PPK similarity (which Jebara et al. 2007 showed to be superior).

HMMs can also be clustered by sampling a number of time-series from each of the HMMs in the base mixture, and then applying the EM algorithm for H3Ms (Smyth, 1997), to cluster the time-series. Despite its simplicity, this approach would suffer from high memory and time requirements, especially when dealing with a large number of input HMMs. First, all generated samples need to be stored in memory. Second, evaluating the likelihood of the generated samples at each iteration is computationally intensive, and prevents the EM algorithm from converging to effective estimates in acceptable times.[6] On the contrary, VHEM-H3M is more efficient in computation and memory usage, as it replaces a costly sampling step (along with the associated likelihood computations at each iteration) with an expectation. An additional problem of EM with sampling is that, with a simple application of the EM algorithm, time-series generated from the same input HMM can be assigned to different clusters of the output H3M. As a consequence, the resulting clustering is not necessary *consistent*, since in this case the corresponding input HMM may not be clearly assigned to any single cluster. In our experiments, we circumvent this problem by defining appropriate constrains on the assignment variables.

The VHEM algorithm is similar in spirit to Bregman-clustering by Banerjee et al. (2005). Both algorithms base clustering on KL-divergence—the KL divergence and the expected

4. The kernel in (29) is computed by marginalizing out the hidden state variables, that is, $\int \left(\sum_{\mathbf{x}} p(\mathbf{y},\mathbf{x}|\mathcal{M}^{(a)})\right)^{\lambda} \left(\sum_{\mathbf{x}} p(\mathbf{y},\mathbf{x}|\mathcal{M}^{(b)})\right)^{\lambda} d\mathbf{y}$. This can be efficiently solved with the junction tree algorithm only when $\lambda = 1$. For $\lambda \neq 1$, Jebara et al. (2004) propose to use an alternative kernel $\tilde{k}$ that applies the power operation to the terms of the sum rather than the entire sum, where the terms are joint probabilities $p(\mathbf{y},\mathbf{x})$. I.e., $\tilde{k}(a,b) = \int \sum_{\mathbf{x}} \left(p(\mathbf{y},\mathbf{x}|\mathcal{M}^{(a)})\right)^{\lambda} \sum_{\mathbf{x}} \left(p(\mathbf{y},\mathbf{x}|\mathcal{M}^{(b)})\right)^{\lambda} d\mathbf{y}$.
5. For two HMM distributions, $\mathcal{M}^{(a)}$ and $\mathcal{M}^{(b)}$, Juang and Rabiner (1985) consider the affinity $L(a,b) = \frac{1}{2}\left[\log p(Y_b|\mathcal{M}^{(a)}) + p(Y_a|\mathcal{M}^{(b)})\right]$, where Y_a and Y_b are sets of observation sequences generated from $\mathcal{M}^{(a)}$ and $\mathcal{M}^{(b)}$, respectively.
6. In our experiments, EM on generated samples took two orders of magnitude more time than VHEM.

log-likelihood differ only for an entropy term that does not affect the clustering.[7] The main differences are: 1) in our setting, the expected log-likelihood (and KL divergence) is not computable in closed form, and hence VHEM uses an approximation; 2) VHEM-H3M clusters random *processes* (i.e., time series models), whereas Bregman-clustering (Banerjee et al., 2005) is limited to single random variables. Note that the number of virtual observations N allows to control the *peakiness* of the assignments $\hat{z}_{i,j}$. Limiting cases for $N \to \infty$ and $N = 1$ are similar to Bregman *hard* and *soft* cluttering, respectively (Goldberger and Roweis, 2004; Banerjee et al., 2005; Dhillon, 2007).

In the next two sections, we validate the points raised in this discussion through experimental evaluation using the VHEM-H3M algorithm. In particular, we consider clustering experiments in Section 5, and H3M density estimation for automatic annotation and retrieval in Section 6. Each application exploits some of the benefits of VHEM. First, we show that VHEM-H3M is more accurate in clustering than PPK-SC, in particular at higher levels of a hierarchical clustering (Section 5.2), and in an experiment with synthetic data (Section 5.3). Similarly, the annotation and retrieval results in Section 6 favor VHEM-H3M over PPK-SC and over standard EM, suggesting that VHEM-H3M is more robust and effective for H3M density estimation. Finally, in all the experiments, the running time of VHEM-H3M compares favorably with the other HMM clustering algorithms; PPK-SC suffers long delays when the number of input HMMs is large and the standard EM algorithm is considerably slower. This demonstrates that VHEM-H3M is most efficient for clustering HMMs.

5. Clustering Experiments

In this section, we present an empirical study of the VHEM-H3M algorithm for clustering and hierarchical clustering of HMMs. Clustering HMMs consists in partitioning K_1 input HMMs into $K_2 < K_1$ groups of similar HMMs. Hierarchical clustering involves organizing the input HMMs in a multi-level hierarchy with h levels, by applying clustering in a recursive manner. Each level ℓ of the hierarchy has K_ℓ groups (with $K_1 > K_2 > \cdots > K_{h-1} > K_h$), and the first level consists of the K_1 input HMMs.

We begin with an experiment on hierarchical clustering, where each of the input HMMs to be clustered is estimated on a sequence of motion capture data (Section 5.2). Then, we present a simulation study on clustering synthetic HMMs (Section 5.3). First, we provide an overview of the different algorithms used in this study.

5.1 Clustering Methods

In the clustering experiments, we will compare our VHEM-H3M algorithm with several other clustering algorithms. The various algorithms are summarized here.

- **VHEM-H3M**: We cluster K_1 input HMMs into K_2 clusters by using the VHEM-H3M algorithm (on the input HMMs) to learn a H3M with K_2 components (as explained

7. We can show that the VHEM algorithm performs clustering based on KL divergence. Letting $\mathcal{D}^{i,j}_{HMM} = \mathcal{L}^{i,i}_{HMM} - \mathcal{L}^{i,j}_{HMM} \approx D(\mathcal{M}^{(b)}_i || \mathcal{M}^{(r)}_j)$ be an approximation to the KL using (21), we can rewrite the E-step as $\hat{z}_{ij} \propto \omega^{(r)}_j e^{-N\omega^{(b)}_i \mathcal{D}^{i,j}_{HMM}}$. Similarly, the M-step is $\hat{\mathcal{M}}^{(r)}_j = \arg\min_{\mathcal{M}^{(r)}_j} \sum_{i=1}^{K^{(b)}} \omega^{(b)}_i \hat{z}_{ij} \mathcal{D}^{i,j}_{HMM}$.

in Section 3.1). To build a multi-level hierarchy of HMMs with h levels, we start from the first level of K_1 input HMMs, and recursively use the VHEM-H3M algorithm $h-1$ times. Each new level ℓ is formed by clustering the $K_{\ell-1}$ HMMs at the previous level into $K_\ell < K_{\ell-1}$ groups with the VHEM-H3M algorithm, and using the learned HMMs as cluster centers at the new level. In our experiments, we set the number of virtual samples to $N = 10^4 K^{(\ell-1)}$, a large value that favors "hard" clustering (where each HMM is univocally assigned to a single cluster), and the length of the virtual sequences to $\tau = 10$.

- **PPK-SC**: Jebara et al. (2007) cluster HMMs by calculating a PPK similarity matrix between all HMMs, and then applying spectral clustering. The work in Jebara et al. (2007) only considered HMMs with single Gaussian emissions, which did not always give satisfactory results in our experiments. Hence, we extended the method of Jebara et al. (2007) by allowing GMM emissions, and derived the PPK similarity for this more general case (Jebara et al., 2004). From preliminary experiments, we found the best performance for PPK with $\lambda = \frac{1}{2}$ (i.e., Bhattacharyya affinity), and when integrating over sequences of length $\tau = 10$. Finally, we also extend Jebara et al. (2007) to construct multi-level hierarchies, by using hierarchical k-means in the spectral clustering embedding.

- **SHEM-H3M**: This is a version of HEM-H3M that maximizes the likelihood of *actual* samples generated from the input HMMs, as in (5), rather than the expectation of virtual samples, as in (6). In particular, from each input HMM $\mathcal{M}_i^{(b)}$ we sample a set Y_i of $N_i = \pi_i^{(b)} N$ observation sequences (for a large value of N). We then estimate the reduced H3M from the N samples $Y = \{Y_i\}_{i=1}^{K^{(b)}}$, with the EM-H3M algorithm of Smyth (1997), which was modified to use a single assignment variable for each sample set Y_i, to obtain a consistent clustering.

In many real-life applications, the goal is to cluster a collection of time series, that is, observed sequences. Although the input data is not a collection of HMMs in that case, it can still be clustered with the VHEM-H3M algorithm by first modeling each sequence as an HMM, and then using the HMMs as input for the VHEM-H3M algorithm. With time-series data as input, it is also possible to use clustering approaches that do not model each sequence as a HMM. Hence, in one of the hierarchical motion clustering experiments, we also compare to the following two algorithms, one that clusters time-series data directly (Smyth, 1997), and a second one that clusters the time series after modeling each sequence with a dynamic texture (DT) model (Chan et al., 2010a).

- **EM-H3M**: The EM algorithm for H3Ms (Smyth, 1997) is applied directly on a collection of time series to learn the clustering and HMM cluster centers, thus bypassing the intermediate HMM modeling stage. To obtain a hierarchical clustering (with $h \geq 3$ levels), we proceed in a bottom up fashion and build each new level by simply re-clustering the given time series in a smaller number of clusters using the EM algorithm by Smyth (1997). We extend the algorithm to use a single assignment variable for each set of sequences Y_i that are within the same cluster in the immediately lower

level of the hierarchy. This modification preserves the hierarchical clustering property that sequences in a cluster will remain together at the higher levels.

- **HEM-DTM**: Rather than use HMMs, we consider a clustering model based on linear dynamical systems, that is, dynamic textures (DTs) (Doretto et al., 2003). Hierarchical clustering is performed using the hierarchical EM algorithm for DT mixtures (HEM-DTM) (Chan et al., 2010a), in an analogous way to VHEM-H3M. The main difference is that, with HEM-DTM, time-series are modeled as DTs, which have a *continuous* state space (a Gauss-Markov model) and *unimodal* observation model, whereas VHEM-H3M uses a *discrete* state space and *multimodal* observations (GMMs).

We will use several metrics to quantitatively compare the results of different clustering algorithms. First, we will calculate the *Rand-index* (Hubert and Arabie, 1985), which measures the correctness of a proposed clustering against a given ground truth clustering. Intuitively, this index measures how consistent cluster assignments are with the ground truth (i.e., whether pairs of items are correctly or incorrectly assigned to the same cluster, or different clusters). Second, we will consider the *log-likelihood*, as used by Smyth (1997) to evaluate a clustering. This measures how well the clustering fits the input data. When time series are given as input data, we compute the log-likelihood of a clustering as the sum of the log-likelihoods of each input sequence under the HMM cluster center to which it has been assigned. When the input data consists of HMMs, we will evaluate the log-likelihood of a clustering by using the expected log-likelihood of observations generated from an input HMM under the HMM cluster center to which it is assigned. For PPK-SC, the cluster center is estimated by running the VHEM-H3M algorithm (with $K^{(r)} = 1$) on the HMMs assigned to the cluster.[8] Note that the log-likelihood will be particularly appropriate to compare VHEM-H3M, SHEM-H3M, EM-H3M and HEM-DTM, since they explicitly optimize for it. However, it may be unfair for PPK-SC, since this method optimizes the PPK similarity and not the log-likelihood. As a consequence, we also measure the *PPK cluster-compactness*, which is more directly related to what PPK-SC optimizes for. The PPK cluster-compactness is the sum (over all clusters) of the average intra-cluster PPK pair-wise similarity. This performance metric favors methods that produce clusters with high intra-cluster similarity.

Note that, *time series* can also be clustered with recourse to similarity measures based on dynamic time warping (Oates et al., 1999; Keogh and Pazzani, 2000; Keogh and Ratanamahatana, 2005) or methods that rely on non-parametric sequence kernels (Leslie et al., 2002; Campbell, 2003; Kuksa et al., 2008; Cortes et al., 2008), which have shown good performance in practice. In this work we focus on the problem of clustering *hidden Markov models*, so we do not pursue an empirical evaluation of these methods.

5.2 Hierarchical Motion Clustering

In this experiment we test the VHEM algorithm on hierarchical motion clustering from motion capture data, that is, time series representing human locomotions and actions. To hierarchically cluster a collection of time series, we first model each time series with an HMM and then cluster the HMMs hierarchically. Since each HMM summarizes the appearance

8. Alternatively, we could use as cluster center the HMM mapped the closest to the spectral embedding cluster center, but this always resulted in lower log-likelihood.

(a) "Walk" sequence.

(b) "Run" sequence.

Figure 1: Examples of motion capture sequences from the MoCap data set, shown with stick figures.

and dynamics of the particular motion sequence it represents, the structure encoded in the hierarchy of HMMs directly applies to the original motion sequences. Jebara et al. (2007) uses a similar approach to cluster motion sequences, applying PPK-SC to cluster HMMs. However, they did not extend their study to hierarchies with multiple levels.

5.2.1 Data Sets and Setup

We experiment on two motion capture data sets, the MoCap data set (`http://mocap.cs.cmu.edu/`) and the Vicon Physical Action data set (Theodoridis and Hu, 2007; Asuncion and Newman, 2010). For the MoCap data set, we use 56 motion examples spanning 8 different classes ("jump", "run", "jog", "walk 1", "walk 2", "basket", "soccer", and "sit"). Each example is a sequence of 123-dimensional vectors representing the (x, y, z)-coordinates of 41 body markers tracked spatially through time. Figure 1 illustrates some typical examples. We built a hierarchy of $h = 4$ levels. The first level (Level 1) was formed by the $K_1 = 56$ HMMs learned from each individual motion example (with $S = 4$ hidden states, and $M = 2$ components for each GMM emission). The next three levels contain $K_2 = 8$, $K_3 = 4$ and $K_4 = 2$ HMMs. We perform the hierarchical clustering with VHEM-H3M, PPK-SC, EM-H3M, SHEM-H3M ($N \in \{560, 2800\}$ and $\tau = 10$), and HEM-DTM (state dimension of 7). The experiments were repeated 10 times for each clustering method, using different random initializations of the algorithms.

The Vicon Physical Action data set is a collection of 200 motion sequences. Each sequence consists of a time series of 27-dimensional vectors representing the (x, y, z)-coordinates of 9 body markers captured using the Vicon 3D tracker. The data set includes 10 normal and 10 aggressive activities, performed by each of 10 human subjects a single time. We build a hierarchy of $h = 5$ levels, starting with $K_1 = 200$ HMMs (with $S = 4$ hidden states and $M = 2$ components for each GMM emission) at the first level (i.e., one for each motion sequence), and using $K_2 = 20$, $K_3 = 8$, $K_4 = 4$, and $K_5 = 2$ for the next four levels. The experiment was repeated 5 times with VHEM-H3M and PPK-SC, using different random initializations of the algorithms.

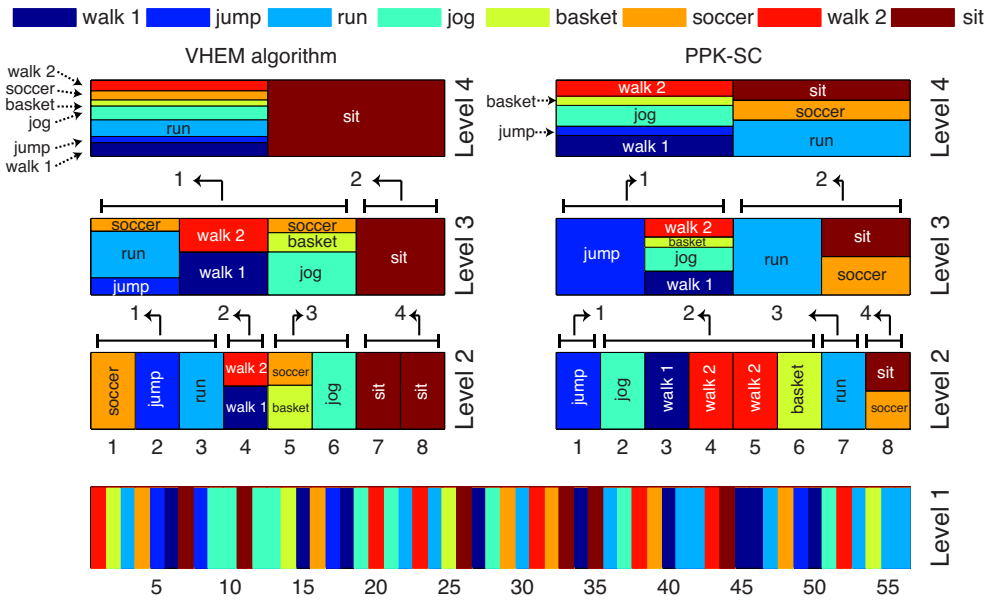

Figure 2: An example of hierarchical clustering of the MoCap data set, with VHEM-H3M and PPK-SC. Different colors represent different motion classes. Vertical bars represent clusters, with the colors indicating the proportions of the motion classes in a cluster, and the numbers on the x-axes representing the clusters' indexes. At Level 1 there are 56 clusters, one for each motion sequence. At Levels 2, 3 and 4 there are 8, 4 and 2 HMM clusters, respectively. For VHEM almost all clusters at Level 2 are populated by examples from a single motion class. The error of VHEM in clustering a portion of "soccer" with "basket" is probably because both actions involve a sequence of movement, shot, and pause. Moving up the hierarchy, the VHEM algorithm clusters similar motions classes together, and at Level 4 creates a dichotomy between "sit" and the other (more dynamic) motion classes. PPK-SC also clusters motion sequences well at Level 2, but incorrectly aggregates "sit" and "soccer", which have quite different dynamics. At Level 4, the clustering obtained by PPK-SC is harder to interpret than that by VHEM.

In similar experiments where we varied the number of levels h of the hierarchy and the number of clusters at each level, we noted similar relative performances of the various clustering algorithms, on both data sets.

5.2.2 Results on the MoCap Data Set

An example of hierarchical clustering of the MoCap data set with VHEM-H3M is illustrated in Figure 2 (left). In the first level, each vertical bar represents a motion sequence, with different colors indicating different ground-truth classes. In the second level, the $K_2 = 8$ HMM clusters are shown with vertical bars, with the colors indicating the proportions of the motion classes in the cluster. Almost all clusters are populated by examples from a single motion class (e.g., "run", "jog", "jump"), which demonstrates that VHEM can

	Rand-index			log-likelihood ($\times 10^6$)			PPK cluster-compactness			time (s)
Level	2	3	4	2	3	4	2	3	4	
VHEM-H3M	0.937	**0.811**	**0.518**	**-5.361**	**-5.682**	**-5.866**	0.0075	**0.0068**	**0.0061**	**30.97**
PPK-SC	**0.956**	0.740	0.393	-5.399	-5.845	-6.068	0.0082	0.0021	0.0008	37.69
SHEM-H3M (560)	0.714	0.359	0.234	-13.632	-69.746	-275.650	0.0062	0.0034	0.0031	843.89
SHEM-H3M (2800)	0.782	0.685	0.480	-14.645	-30.086	-52.227	0.0050	0.0036	0.0030	3849.72
EM-H3M	0.831	0.430	0.340	-5.713	-202.55	-168.90	**0.0099**	0.0060	0.0056	667.97
HEM-DTM	0.897	0.661	0.412	-7.125	-8.163	-8.532	-	-	-	121.32

Table 2: Hierarchical clustering of the MoCap data set using VHEM-H3M, PPK-SC, SHEM-H3M, EM-H3M and HEM-DTM. The number in brackets after SHEM-H3M represents the number of real samples used. We computed Rand-index, data log-likelihood and cluster compactness at each level of the hierarchy, and registered the time (in seconds) to learn the hierarchical structure. Differences in Rand-index at Levels 2, 3, and 4 are statistically significant based on a paired t-test with confidence 95%.

group similar motions together. We note an error of VHEM in clustering a portion of the "soccer" examples with "basket". This is probably caused by the fact that both types of actions begin with a stationary phase (e.g., subject focusing on the execution) followed with a forward movement (note that our "basket" examples correspond to forward dribbles). Moving up the hierarchy, the VHEM algorithm clusters similar motion classes together (as indicated by the arrows), for example "walk 1" and "walk 2" are clustered together at Level 2, and at the highest level (Level 4) it creates a dichotomy between "sit" and the rest of the motion classes. This is a desirable behavior as the kinetics of the "sit" sequences (which in the MoCap data set correspond to starting in a standing position, sitting on a stool, and returning to a standing position) are considerably different from the rest. On the right of Figure 2, the same experiment is repeated with PPK-SC. PPK-SC clusters motion sequences properly, but incorrectly aggregates "sit" and "soccer" at Level 2, even though they have quite different dynamics. Furthermore, the highest level (Level 4) of the hierarchical clustering produced by PPK-SC is harder to interpret than that of VHEM.

Table 2 presents a quantitative comparison between PPK-SC and VHEM-H3M at each level of the hierarchy. While VHEM-H3M has lower Rand-index than PPK-SC at Level 2 (0.937 vs. 0.956), VHEM-H3M has higher Rand-index at Level 3 (0.811 vs. 0.740) and Level 4 (0.518 vs. 0.393). In terms of PPK cluster-compactness, we observe similar results. In particular, VHEM-H3M has higher PPK cluster-compactness than PPK-SC at Level 3 and 4. Overall, keeping in mind that PPK-SC is explicitly driven by PPK-similarity, while the VHEM-H3M algorithm is not, these results can be considered as strongly in favor of VHEM-H3M (over PPK-SC). In addition, the data log-likelihood for VHEM-H3M is higher than that for PPK-SC at each level of the hierarchy. This suggests that the novel HMM cluster centers learned by VHEM-H3M fit the motion capture data better than the spectral cluster centers, since they condense information of the entire underlying clusters. This conclusion is further supported by the results of the density estimation experiments in Sections 6.1 and 6.2. Note that the higher up in the hierarchy, the more clearly this effect is manifested.

Comparing to other methods (also in Table 2), EM-H3M generally has lower Rand-index than VHEM-H3M and PPK-SC (consistent with the results from Jebara et al. (2007)). While EM-H3M directly clusters the original motion sequences, both VHEM-H3M and PPK-SC implicitly integrate over all possible virtual variations of the original motion sequences (according to the intermediate HMM models), which results in more robust clustering procedures. In addition, EM-H3M has considerably longer running times than VHEM-H3M and PPK-SC (i.e., roughly 20 times longer) since it needs to evaluate the likelihood of all training sequences at each iteration, at all levels.

The results in Table 2 favor VHEM-H3M over SHEM-H3M, and empirically validate the variational approximation that VHEM uses for learning. For example, when using $N = 2800$ samples, running SHEM-H3M takes over two orders of magnitude more time than VHEM-H3M, but still does not achieve performance competitive with VHEM-H3M. With an efficient closed-form expression for averaging over all possible virtual samples, VHEM approximates the sufficient statistics of a virtually unlimited number of observation sequences, without the need of using real samples. This has an additional regularization effect that improves the robustness of the learned HMM cluster centers. In contrast, SHEM-H3M uses real samples, and requires a large number of them to learn accurate models, which results in significantly longer running times.

In Table 2, we also report hierarchical clustering performance for HEM-DTM. VHEM-H3M consistently outperforms HEM-DTM, both in terms of Rand-index and data log-likelihood.[9] Since both VHEM-H3M and HEM-DTM are based on the hierarchical EM algorithm for learning the clustering, this indicates that HMM-based clustering models are more appropriate than DT-based models for the human MoCap data. Note that, while PPK-SC is also HMM-based, it has a lower Rand-index than HEM-DTM at Level 4. This further suggests that PPK-SC does not optimally cluster the HMMs.

Finally, to assess stability of the clustering results, starting from the 56 HMMs learned on the motion examples, in turn we exclude one of them and build a hierarchical clustering of the remaining ones ($h = 4$ levels, $K_1 = 55, K_2 = 8, K_3 = 4, K_4 = 2$), using in turn VHEM-H3M and PPK-SC. We then compute cluster stability as the mean Rand-index between all possible pairs of clusterings. The experiment is repeated 10 times for both VHEM-H3M and PPK-SC, using a different random initialization for each trial. The stability of VHEM-H3M is 0.817, 0.808 and 0.950, at Levels 2, 3 and 4. The stability of PPK-SC is 0.786, 0.837 and 0.924, at Levels 2, 3 and 4. Both methods are fairly stable in terms of Rand-index, with a slight advantage for VHEM-H3M over PPK-SC (with average across levels of 0.858 versus 0.849).

The experiments were repeated 10 times for each clustering method, using different random initializations of the algorithms.

Alternatively, we evaluate the out-of-sample generalization of the clusterings discovered by VHEM-H3M and PPK-SC, by computing the fraction of times the held out HMMs is assigned[10] to the cluster that contains the majority of HMMs from its same ground truth

9. We did not report PPK cluster-compactness for HEM-DTM, since it would not be directly comparable with the same metric based on HMMs.

10. For VHEM-H3M we use the expected log-likelihood, for PPK-SC we use the out-of-sample extension for spectral clustering of Bengio et al. (2004).

	Rand-index				log-likelihood ($\times 10^6$)				PPK cluster-compactness			
Level	2	3	4	5	2	3	4	5	2	3	4	5
VHEM-H3M	0.909	0.805	0.610	0.244	-1.494	-3.820	-5.087	-6.172	0.224	0.059	0.020	0.005
PPK-SC	0.900	0.807	0.452	0.092	-3.857	-5.594	-6.163	-6.643	0.324	0.081	0.026	0.008

Table 3: Hierarchical clustering of the Vicon Physical Action data set using VHEM-H3M and PPK-SC. Performance is measured in terms of Rand-index, data log-likelihood and PPK cluster-compactness at each level. Differences in Rand-index at Levels 2, 4 and 5 are statistically significant based on a paired t-test with confidence 95%. The test failed at Level 3.

class. Out of sample generalization of VHEM-H3M and PPK-SC are comparable, registering an averages of 0.875, respectively, 0.851 across the different levels.

5.2.3 Results on the Vicon Physical Action Data Set

Table 3 presents results using VHEM-H3M and PPK-SC to cluster the Vicon Physical Action data set. While the two algorithms performs similarly in terms of Rand-index at lower levels of the hierarchy (i.e., Level 2 and Level 3), at higher levels (i.e., Level 4 and Level 5) VHEM-H3M outperforms PPK-SC. In addition, VHEM-H3M registers higher data log-likelihood than PPK-SC at each level of the hierarchy. This, again, suggests that by learning new cluster centers, the VHEM-H3M algorithm retains more information on the clusters' structure than PPK-SC. Finally, compared to VHEM-H3M, PPK-SC produces clusters that are more compact in terms of PPK similarity. However, this does not necessarily imply a better agreement with the ground truth clustering, as evinced by the Rand-index metrics.

5.3 Clustering Synthetic Data

In this experiment, we compare VHEM-H3M and PPK-SC on clustering a synthetic data set of HMMs.

5.3.1 Data Set and Setup

The synthetic data set of HMMs is generated as follows. Given a set of C HMMs $\{\mathcal{M}^{(c)}\}_{c=1}^{C}$, for each HMM we synthesize a set of K "noisy" versions of the original HMM. Each "noisy" HMM $\tilde{\mathcal{M}}_k^{(c)}$ $(k = 1, \dots K)$ is synthesized by generating a random sequence $y_{1:T}$ of length T from $\mathcal{M}^{(c)}$, corrupting it with Gaussian noise $\sim \mathcal{N}(0, \sigma_n^2 \mathrm{I}_d)$, and estimating the parameters of $\tilde{\mathcal{M}}_k^{(c)}$ on the corrupted version of $y_{1:T}$. Note that this procedure adds noise in the *observation* space. The number of noisy versions (of each given HMM), K, and the noise variance, σ_n^2, will be varied during the experiments.

The collection of original HMMs was created as follows. Their number was always set to $C = 4$, the number of hidden states of the HMMs to $S = 3$, the emission distributions to be single, one-dimensional Gaussians (i.e., GMMs with $M = 1$ component), and the length of the sequences to $T = 100$. For all original HMMs $\mathcal{M}^{(c)}$, *if not otherwise specified*, the initial state probability, the state transition matrix, and the means and variances of the

emission distributions were fixed as

$$\pi^{(c)} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \quad A^{(c)} = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.2 & 0.8 & 0 \\ 0 & 0.2 & 0.8 \end{bmatrix}, \quad \begin{cases} \mu_1^{(c)} = 1 \\ \mu_2^{(c)} = 2 \\ \mu_3^{(c)} = 3 \end{cases} \quad {\sigma_\rho^{(c)}}^2 = 0.5 \ \forall \rho, \quad \forall c.$$

We consider three different experimental settings. In the first setting, experiment (a), the HMMs $\mathcal{M}^{(c)}$ only differ in the means of the emission distributions,

$$\begin{cases} \mu_1^{(1)} = 1 \\ \mu_2^{(1)} = 2 \\ \mu_3^{(1)} = 3 \end{cases}, \begin{cases} \mu_1^{(2)} = 3 \\ \mu_2^{(2)} = 2 \\ \mu_3^{(2)} = 1 \end{cases}, \begin{cases} \mu_1^{(3)} = 1 \\ \mu_2^{(3)} = 2 \\ \mu_3^{(3)} = 2 \end{cases}, \begin{cases} \mu_1^{(4)} = 1 \\ \mu_2^{(4)} = 3 \\ \mu_3^{(4)} = 3 \end{cases}$$

In the second setting, experiment (b), the HMMs differ in the variances of the emission distributions,

$${\sigma_\rho^{(1)}}^2 = 0.5, \quad {\sigma_\rho^{(2)}}^2 = 0.1, \quad {\sigma_\rho^{(3)}}^2 = 1, \quad {\sigma_\rho^{(4)}}^2 = 0.05, \quad \forall \rho.$$

In the last setting, experiment (c), the HMMs differ in the transition matrices,

$$A^{(1)} = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.2 & 0.8 & 0 \\ 0 & 0.2 & 0.8 \end{bmatrix} \ A^{(2)} = \begin{bmatrix} 0.2 & 0.2 & 0.2 \\ 0.4 & 0.6 & 0 \\ 0 & 0.4 & 0.6 \end{bmatrix} \ A^{(3)} = \begin{bmatrix} 0.9 & 0.05 & 0.05 \\ 0.1 & 0.9 & 0 \\ 0 & 0.1 & 0.9 \end{bmatrix} \ A^{(4)} = \begin{bmatrix} 0.4 & 0.3 & 0.4 \\ 0.6 & 0.4 & 0 \\ 0 & 0.6 & 0.4 \end{bmatrix}.$$

The VHEM-H3M and PPK-SC algorithms are used to cluster the synthesized HMMs, $\{\{\tilde{\mathcal{M}}_k^{(c)}\}_{k=1}^K\}_{c=1}^C$, into C groups, and the quality of the resulting clusterings is measured with the Rand-index, PPK cluster-compactness, and the expected log-likelihood of the discovered cluster centers with respect to the original HMMs. The expected log-likelihood was computed using the lower bound, as in (13), with each of the original HMMs assigned to the most likely HMM cluster center. The results are averages over 10 trials.

The reader is referred to Appendix E for experiments where the order of the model used for clustering does not match the order of the true model used for generating the data.

5.3.2 Results

Figure 3 reports the performance metrics when varying the number $K \in \{2, 4, 8, 16, 32\}$ of noisy versions of each of the original HMMs, and the noise variance $\sigma_n^2 \in \{0.1, 0.5, 1\}$, for the three experimental settings. (Note that, in each trial, for each class, we first generated 32 noisy HMMs per class, and then, varying $K \in \{4, 8, 16, 32\}$, we subsampled only K of them.) For the majority of settings of K and σ_n^2, the clustering produced by VHEM-H3M is superior to the one produced by PPK-SC, for each of the considered metrics (i.e., in the plots, solid lines are usually above dashed lines of the same color). The only exception is in experiment (b) where, for low noise variance (i.e., $\sigma_n^2 = 0.1$) PPK-SC is the best in terms of Rand-index and cluster compactness. It is interesting to note that the gap in performance between VHEM-H3M and PPK-SC is generally larger at low values of K. We believe this is because, when only a limited number of input HMMs is available, PPK-SC produces an

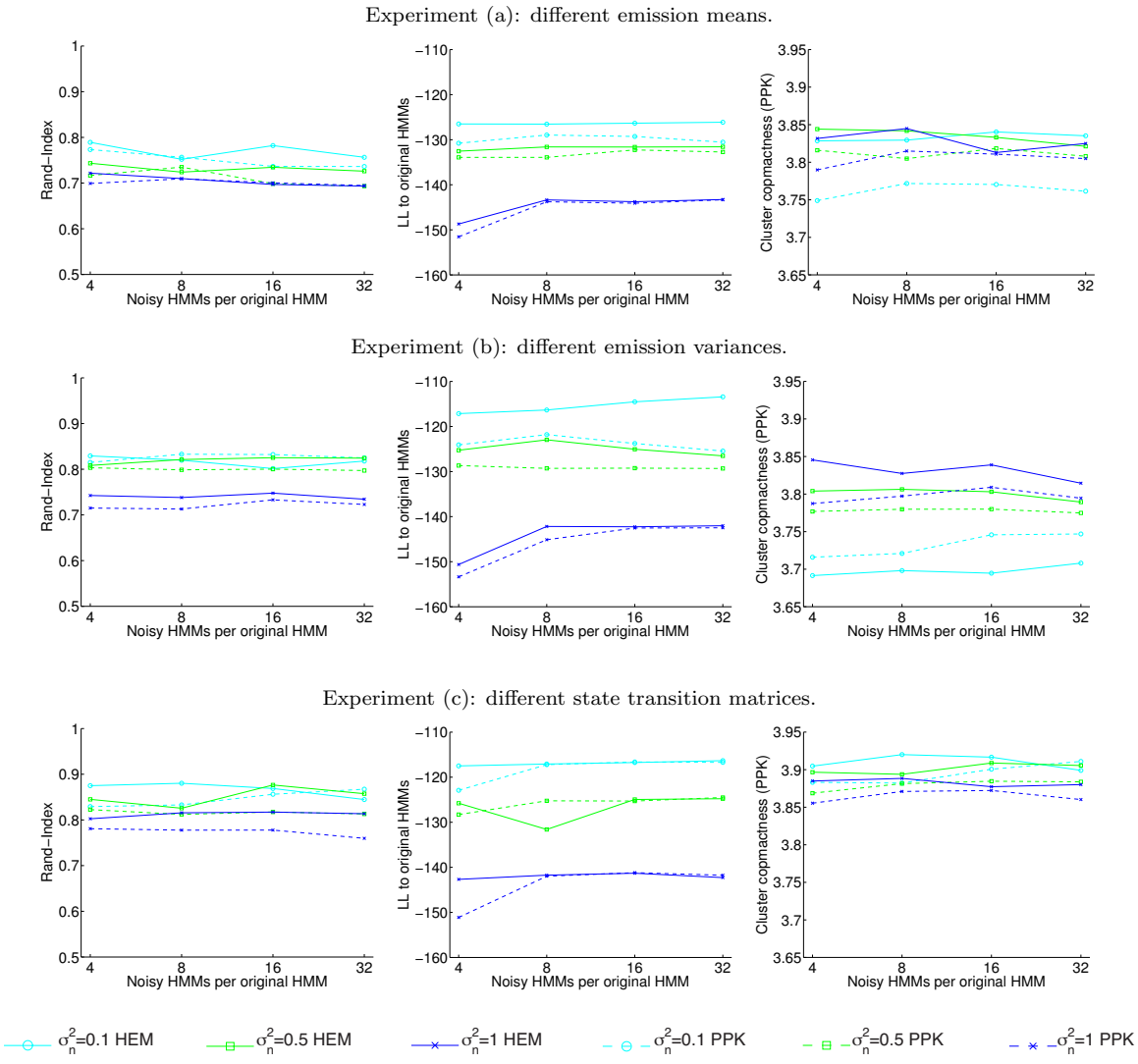

Figure 3: Results on clustering synthetic data with VHEM-H3M and PPK-SC. Performance is measured in terms of Rand-index, expected log-likelihood and PPK cluster-compactness.

embedding of lower quality. This does not affect VHEM-H3M, since it clusters in HMM distribution space and does not use an embedding.

Note that the Rand-Index values for experiment (c) (i.e., different state transition matrices) are superior to the corresponding ones in experiments (a) and (b) (i.e., different emission distributions) for both VHEM-H3M and PPK-SC. This shows that VHEM-H3M (and slightly less robustly PPK-SC as well) can cluster dynamics characterized by different hidden states processes more easily than dynamics that differ only in the emission distri-

butions. In addition, VHEM-H3M is more robust to noise than PPK-SC, as demonstrated by the experiments with $\sigma_n^2 = 1$ (blue lines).

These results suggest that, by clustering HMMs *directly* in distribution space, VHEM-H3M is generally more robust than PPK-SC, the performance of which instead depends on the quality of the underlying embedding.

Finally, the results in Figure 3 show only small fluctuations across different values of K at each noise value, suggesting stability of the clustering results (*relative to the ground truth clustering*) for both VHEM-H3M and PPK-SC, to both subsampling and jittering (i.e., addition of noise) (Hennig, 2007). In particular, from the expected log-likelihood values (central column in Figure 3), we evince that the similarity of the discovered clustering to the true distribution (i.e., the original HMMs) is not largely affected by the amount of subsampling, except when only as little as the 12.5% of the data is used (when $K = 4$).

6. Density Estimation Experiments

In this section, we present an empirical study of VHEM-H3M for density estimation, in automatic annotation and retrieval of music (Section 6.1) and hand-written digits (Section 6.2).

6.1 Music Annotation and Retrieval

In this experiment, we evaluate VHEM-H3M for estimating annotation models in content-based music auto-tagging. As a generative time-series model, H3Ms allow to account for timbre (i.e., through the GMM emission process) as well as longer term temporal dynamics (i.e., through the HMM hidden state process), when modeling musical signals. Therefore, in music annotation and retrieval applications, H3Ms are expected to prove more effective than existing models that do not explicitly account for temporal information (Turnbull et al., 2008; Mandel and Ellis, 2008; Eck et al., 2008; Hoffman et al., 2009).

6.1.1 Music Data Set

We consider the CAL500 collection from Turnbull et al. (2008), which consists of 502 songs and provides binary annotations with respect to a vocabulary $\mathcal{V}$ of 149 tags, ranging from genre and instrumentation, to mood and usage. To represent the acoustic content of a song we extract a time series of audio features $\mathcal{Y} = \{y_1, \ldots, y_{|\mathcal{Y}|}\}$, by computing the first 13 Mel frequency cepstral coefficients (MFCCs) (Rabiner and Juang, 1993) over half-overlapping windows of 46ms of audio signal, augmented with first and second instantaneous derivatives. The song is then represented as a collection of *audio fragments*, which are sequences of $T = 125$ audio features (approximately 6 seconds of audio), using a dense sampling with 80% overlap.

6.1.2 Music Annotation Models

Automatic music tagging is formulated as a supervised multi-label problem (Carneiro et al., 2007), where each class is a tag from $\mathcal{V}$. We approach this problem by modeling the audio content for each tag with a H3M probability distribution. I.e., for each tag, we estimate an H3M over the audio fragments of the songs in the database that have been associated

with that tag, using the hierarchical estimation procedure based on VHEM-H3M. More specifically, the database is first processed at the song level, using the EM algorithm to learn a H3M with $K^{(s)} = 6$ components for each song[11] from its audio fragments. Then, for each tag, the song-level H3Ms labeled with that tag are pooled together to form a large H3M, and the VHEM-H3M algorithm is used to reduce this to a final H3M tag-model with $K = 3$ components ($\tau = 10$ and $N = N_v N_t K^{(s)}$, where $N_v = 1000$ and N_t is the number of training songs for the particular tag).

Given the tag-level models, a song can be represented as a vector of posterior probabilities of each tag (a semantic multinomial, SMN), by extracting features from the song, computing the likelihood of the features under each tag-level model, and applying Bayes' rule.[12] A test song is annotated with the top-ranking tags in its SMN. To retrieve songs given a tag query, a collection of songs is ranked by the tag's probability in their SMNs.

We compare VHEM-H3M with three alternative algorithms for estimating the H3M tag models: PPK-SC, PPK-SC-hybrid, and EM-H3M.[13] For all three alternatives, we use the same number of mixture components in the tag models ($K = 3$). For the two PPK-SC methods, we leverage the work of Jebara et al. (2007) to learn H3M tag models, and use it in place of the VHEM-H3M algorithm in the second stage of the hierarchical estimation procedure. We found that it was necessary to implement the PPK-SC approaches with song-level H3Ms with only $K^{(s)} = 1$ component (i.e., a single HMM), since the computational cost for constructing the initial embedding scales poorly with the number of input HMMs.[14] PPK-SC first applies spectral clustering to the song-level HMMs and then selects as the cluster centers the HMMs that map closest to the spectral cluster centers in the spectral embedding. PPK-SC-hybrid is a hybrid method combining PPK-SC for clustering, and VHEM-H3M for estimating the cluster centers. Specifically, after spectral clustering, HMM cluster centers are estimated by applying VHEM-H3M (with $K^{(r)} = 1$) to the HMMs assigned to each of the resulting clusters. In other words, PPK-SC and PPK-SC-hybrid use spectral clustering to summarize a collection of song-level HMMs with a few HMM centers, forming a H3M tag model. The mixture weight of each HMM component (in the H3M tag model) is set proportional to the number of HMMs assigned to that cluster.

For EM-H3M, the H3M tag models were estimated directly from the audio fragments from the relevant songs using the EM-H3M algorithm.[15] Empirically, we found that, due

11. Most pop songs have 6 structural parts: intro, verse, chorus, solo, bridge and outro.
12. We compute the likelihood of a song under a tag model as the geometric average of the likelihood of the individual segments, and further normalized it by the length of the segments to prevent the posteriors from being too "peaked" (Coviello et al., 2011).
13. For this experiment, we were not able to successfully estimate accurate H3M tag models with SHEM-H3M. In particular, SHEM-H3M requires generating an appropriately large number of real samples to produce accurate estimates. However, due to the computational limits of our cluster, we were able to test SHEM-H3M only using a small number of samples. In preliminary experiments we registered performance only slightly above chance level and training times still twice longer than for VHEM-H3M. For a comparison between VHEM-H3M and SHEM-H3M on density estimation, the reader can refer to the experiment in Section 6.2 on online hand-writing classification and retrieval.
14. Running PPK-SC with $K^{(s)} = 2$ took 3958 hours in total (about 4 times more than when setting $K^{(s)} = 1$), with no improvement in annotation and retrieval performance. A larger $K^{(s)}$ would yield impractically long learning times.
15. The EM algorithm has been used to estimate HMMs from music data in previous work (Scaringella and Zoia, 2005; Reed and Lee, 2006).

to its runtime and RAM requirements, for EM-H3M we must use non-overlapping audio-fragments and evenly subsample by 73% on average, resulting in 14.5% of the sequences used by VHEM-H3M. Note that, however, EM-H3M is still using 73% of the actual song data (just with non-overlapping sequences). We believe this to be a reasonable comparison between EM and VHEM, as both methods use roughly similar resources (the sub-sampled EM is still 3 times slower, as reported in Table 4). Based on our projections, running EM over densely sampled song data would require roughly 9000 hours of CPU time (e.g., more than 5 weeks when parallelizing the algorithm over 10 processors), as opposed to 630 hours for VHEM-H3M. This would be extremely cpu-intensive given the computational limits of our cluster. The VHEM algorithm, on the other hand, can learn from considerable amounts of data while still maintaining low runtime and memory requirements.[16]

Finally, we also compare against two state-of-the-art models for music tagging, HEM-DTM (Coviello et al., 2011), which is based on a different time-series model (mixture of dynamic textures), and HEM-GMM (Turnbull et al., 2008), which is a bag-of-features model using GMMs. Both methods use efficient hierarchical estimation based on a HEM algorithm (Chan et al., 2010a; Vasconcelos and Lippman, 1998) to obtain the tag-level models.[17]

6.1.3 Performance Metrics

A test song is annotated with the 10 most likely tags, and annotation performance is measured with the per-tag precision (P), recall (R), and F-score (F), averaged over all tags. If $|w_H|$ is the number of test songs that have the tag w in their ground truth annotations, $|w_A|$ is the number of times an annotation system uses w when automatically tagging a song, and $|w_C|$ is the number of times w is correctly used, then precision, recall and F-score for the tag w are defined as:

$$\mathrm{P} = \frac{|w_C|}{|w_A|}, \;\; \mathrm{R} = \frac{|w_C|}{|w_H|}, \;\; \mathrm{F} = 2\left((\mathrm{P})^{-1} + (\mathrm{R})^{-1}\right)^{-1}.$$

Retrieval is measured by computing per-tag mean average precision (MAP) and precision at the first k retrieved songs (P@k), for $k \in \{5, 10, 15\}$. The P@k is the fraction of true positives in the top-k of the ranking. MAP averages the precision at each point in the ranking where a song is correctly retrieved. All reported metrics are averages over the 97 tags that have at least 30 examples in CAL500 (11 genre, 14 instrument, 25 acoustic quality, 6 vocal characteristics, 35 emotion and 6 usage tags), and are the result of 5-fold cross-validation.

Finally, we also record the total time (in hours) to learn the 97 tag-level H3Ms on the 5 splits of the data. For hierarchical estimation methods (VHEM-H3M and the PPK-SC approaches), this also includes the time to learn the song-level H3Ms.

16. For example, consider learning a tag-level H3M from 200 songs, which corresponds to over 3GB of audio fragments. Using the hierarchical estimation procedure, we first model each song (in average, 15MB of audio fragments) individually as a song-level H3M, and we save the song models (150 KB of memory each). Then, we pool the 200 song models into a large H3M (in total 30MB of memory), and reduce it to a smaller tag-level H3M using the VHEM-H3M algorithm.

17. Both auto-taggers operate on audio features extracted over half-overlapping windows of 46ms. HEM-GMM uses MFCCs with first and second instantaneous derivatives (Turnbull et al., 2008). HEM-DTM uses 34-bins of Mel-spectral features (Coviello et al., 2011), which are further grouped in audio fragments of 125 consecutive features.

	annotation			retrieval				
	P	R	F	MAP	P@5	P@10	P@15	time (h)
VHEM-H3M	0.446	0.211	0.260	0.440	0.474	0.451	0.435	629.5
PPK-SC	0.299	0.159	0.151	0.347	0.358	0.340	0.329	974.0
PPK-SC-*hybrid*	0.407	0.200	0.221	0.415	0.439	0.421	0.407	991.7
EM-H3M	0.415	0.214	0.248	0.423	0.440	0.422	0.407	1860.4
HEM-DTM	0.431	0.202	0.252	0.439	0.479	0.454	0.428	-
HEM-GMM	0.374	0.205	0.213	0.417	0.441	0.425	0.416	-

Table 4: Annotation and retrieval performance on CAL500, for VHEM-H3M, PPK-SC, PPK-SC-*hybrid*, EM-H3M, HEM-DTM (Coviello et al., 2011) and HEM-GMM (Turnbull et al., 2008).

6.1.4 Results

In Table 4 we report the performance of the various algorithms for both annotation and retrieval on the CAL500 data set. Looking at the overall runtime, VHEM-H3M is the most efficient algorithm for estimating H3M distributions from music data, as it requires only 34% of the runtime of EM-H3M, and 65% of the runtime of PPK-SC. The VHEM-H3M algorithm capitalizes on the first stage of song-level H3M estimation (about one third of the total time) by efficiently and effectively using the song-level H3Ms to learn the final tag models. Note that the runtime of PPK-SC corresponds to setting $K^{(s)} = 1$. When we set $K^{(s)} = 2$, we registered a running time four times longer, with no significant improvement in performance.

The gain in computational efficiency does not negatively affect the quality of the corresponding models. On the contrary, VHEM-H3M achieves better performance than EM-H3M,[18] strongly improving the top of the ranked lists, as evinced by the higher P@k scores. Relative to EM-H3M, VHEM-H3M has the benefit of regularization, and during learning can efficiently leverage all the music data condensed in the song H3Ms. VHEM-H3M also outperforms both PPK-SC approaches on all metrics. PPK-SC discards considerable information on the clusters' structure by selecting one of the original HMMs to approximate each cluster. This significantly affects the accuracy of the resulting annotation models. VHEM-H3M, on the other hand, generates novel HMM cluster centers to summarize the clusters. This allows to retain more accurate information in the final annotation models.

PPK-SC-hybrid achieves considerable improvements relative to standard PPK-SC, at relatively low additional computational costs.[19] This further demonstrates that the VHEM-H3M algorithm can effectively summarize in a smaller model the information contained in *several* HMMs. In addition, we observe that VHEM-H3M still outperforms PPK-SC-hybrid, suggesting that the former produces more accurate cluster centers and density estimates.

18. The differences in performance are statistically significant based on a paired t-test with 95% confidence.
19. In PPK-SC-hybrid, each run of the VHEM-H3M algorithm converges quickly since there is only one HMM component to be learned, and can benefit from clever initialization (i.e., to the HMM mapped the closest to the spectral clustering center).

In fact, VHEM-H3M *couples* clustering and learning HMM cluster centers, and is entirely based on maximum likelihood for estimating the H3M annotation models. PPK-SC-hybrid, on the contrary, separates clustering and parameter estimation, and optimizes them against two different metrics (i.e., PPK similarity and expected log-likelihood, respectively). As a consequence, the two phases may be mismatched, and the centers learned with VHEM may not be the best representatives of the clusters according to PPK affinity.

Finally, VHEM-H3M compares favorably to the auto-taggers based on other generative models. First, VHEM-H3M outperforms HEM-GMM, which does not model temporal information in the audio signal, on all metrics. Second, the performances of VHEM-H3M and HEM-DTM (a continuous-state temporal model) are not statistically different based on a paired t-test with 95% confidence, except for annotation precision where VHEM-H3M scores significantly higher. Since HEM-DTM is based on linear dynamic systems (a continuous-state model), it can model stationary time-series in a linear subspace. In contrast, VHEM-H3M uses HMMs with discrete states and GMM emissions, and can hence better adapt to non-stationary time-series on a non-linear manifold. This difference is illustrated in the experiments: VHEM-H3M outperforms HEM-DTM on the human MoCap data (see Table 2), which has non-linear dynamics, while the two perform similarly on the music data (see Table 4), where audio features are often stationary over short time frames.

6.2 On-Line Hand-Writing Data Classification and Retrieval

In this experiment, we investigate the performance of the VHEM-H3M algorithm in estimating class-conditional H3M distributions for automatic classification and retrieval of on-line hand-writing data.

6.2.1 Data Set

We consider the Character Trajectories Data Set (Asuncion and Newman, 2010), which is a collection of 2858 examples of characters from the same writer, originally compiled to study handwriting motion primitives (Williams et al., 2006).[20] Each example is the trajectory of one of the 20 different characters that correspond to a single pen-down segment. The data was captured from a WACOM tablet at 200 Hz, and consists of (x, y)-coordinates and pen tip force. The data has been numerically differentiated and Gaussian smoothed (Williams et al., 2006). Half of the data is used for training, with the other half held out for testing.

6.2.2 Classification Models and Setup

From the hand-writing examples in the training set, we estimate a series of class-conditional H3M distributions, one for each character, using hierarchical estimation with the VHEM-H3M algorithm. First, for each character, we partition all the relevant training data into groups of 3 sequences, and learn a HMM (with $S = 4$ states and $M = 1$ component for the GMM emissions) from each group using the Baum-Welch algorithm. Next, we estimate the class-conditional distribution (classification model) for each character by aggregating all the relevant HMMs and summarizing them into a H3M with $K = 4$ components using the

20. Even if we limit this experiment to the data set of Williams et al. (2006), we would like to refer the interest readers to the data set of Liwicki and Bunke (2005), which is a collection of *word* trajectories (as opposed to *character* trajectories).

VHEM-H3M algorithm ($\tau = 10$ and $N = N_v N_c$, where $N_v = 10$,[21] and N_c is the number of intermediate models for that character). Using the character-level H3Ms and Bayes' rule, for each hand-writing example in the test set we compute the posterior probabilities of all of the 20 characters. Each example is classified as the character with largest posterior probability. For retrieval given a query character, examples in the test set are ranked by the character's posterior probability.

We repeated the same experiment using PPK-SC or SHEM-H3M ($\tau = 10$, $N = 1000$) to estimate the classification models from the intermediate HMMs. Finally, we considered the EM-H3M algorithm, which directly uses the training sequences to learn the class-conditional H3M ($K = 4$).

Since VHEM-H3M, SHEM-H3M and EM-H3M are *iterative* algorithms, we studied them when varying the stopping criterion. In particular, the algorithms were terminated when the relative variation in the value of the objective function between two consecutive iterations was lower than a threshold ΔLL, which we varied in $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.[22]

Finally, we measure classification and retrieval performance on the test set using the classification accuracy, and the average per-tag P@3, P@5 and MAP. We also report the total training time, which includes the time used to learn the intermediate HMMs. The experiments consisted of 5 trials with different random initializations of the algorithms.

6.2.3 Results

Table 5 lists the classification and retrieval performance on the test set for the various methods. Consistent with the experiments on music annotation and retrieval (Section 6.1), VHEM-H3M performs better than PPK-SC on all metrics. By learning novel HMM cluster centers, VHEM-H3M estimates H3M distributions that are representative of all the relevant intermediate HMMs, and hence of all the relevant training sequences. While EM-H3M is the best in classification (at the price of longer training times), VHEM-H3M performs better in retrieval, as evinced by the P@3 and P@5 scores. In terms of training time, VHEM-H3M and PPK-SC are about 5 times faster than EM-H3M. In particular, PPK-SC is the fastest algorithm, since the small number of input HMMs (i.e., on average 23 per character) allows to build the spectral clustering embedding efficiently.

The version of HEM based on actual sampling (SHEM-H3M) performs better than VHEM-H3M in classification, but VHEM-H3M has higher retrieval scores. However, the training time for SHEM-H3M is approximately 15 times longer than for VHEM-H3M. These differences in performance can be understood in terms of the different types of *approximation* used by SHEM-H3M and VHEM-H3M. The sampling operation of SHEM-H3M provides an *unbiased* estimator, whose *variance* depends on the number of samples used. Hence, in order to reliably estimate the reduced model (i.e., making the variance small), the SHEM-

21. Note that choosing a lower value of N_v (compared to the music experiments) plays a role in making the clustering algorithm more *reliable*. Using fewer virtual samples equates to attaching smaller "virtual probability masses" to the input HMMs, and leads to *less certain* assignments of the input HMMs to the clusters (cf. Equation 19). This determines more mixing in the initial iterations of the algorithm (e.g., similar to higher annealing temperature), and reduces the risk of prematurely specializing any cluster to one of the original HMMs. This effect is desirable, since the input HMMs are estimated over a smaller number of sequences (compared to the music experiments) and can therefore be noisier and less reliable.
22. In a similar experiment where we used the number of iterations as the stopping criterion, we registered similar results.

	stop	**retrieval**			**classification**	
	ΔLL	P@3	P@5	MAP	Accuracy	total time (s)
VHEM-H3M	10^{-2}	0.750	0.750	0.738	0.618	1838.34
	10^{-3}	0.717	0.750	0.741	0.619	1967.55
	10^{-4}	0.733	0.790	0.773	0.648	2210.77
	10^{-5}	0.750	0.820	0.775	0.651	2310.93
SHEM-H3M	10^{-2}	0.417	0.440	0.517	0.530	13089.32
	10^{-3}	0.683	0.680	0.728	0.664	23203.44
	10^{-4}	0.700	0.750	0.753	0.689	35752.20
	10^{-5}	0.700	0.750	0.754	0.690	50094.36
EM-H3M	10^{-2}	0.583	0.610	0.667	0.646	6118.53
	10^{-3}	0.617	0.650	0.731	0.674	7318.56
	10^{-4}	0.650	0.710	0.756	0.707	9655.08
	10^{-5}	0.517	0.560	0.665	0.635	10957.38
PPK-SC	-	0.600	0.700	0.698	0.646	1463.54

Table 5: Classification and retrieval performance, and training time on the Character Trajectories Data Set, for VHEM-H3M, PPK-SC, SHEM-H3M, and EM-H3M.

H3M algorithm requires generating and handling a sufficiently large number of samples (relative to the model order/complexity, Hastie et al. 2005). In contrast, the variational approximation of VHEM-H3M does not require generating samples (and hence avoids the problem of large variances associate to relatively small samples), but introduces a bias in the estimator (Gunawardana and Byrne, 2005). Hence, it is not surprising that, on a relatively less complex problem where a relatively smaller sample size suffices, SHEM-H3M can perform more robustly than VHEM-H3M (even if still at a larger computational cost).

It is also interesting to note that EM-H3M appears to suffer from overfitting of the training set, as suggested by the *overall drop* in performance when the stopping criterion changes from $\Delta LL = 10^{-4}$ to $\Delta LL = 10^{-5}$. In contrast, both VHEM-H3M and SHEM-H3M consistently improve on all metrics as the algorithm converges (again looking at $\Delta LL \in \{10^{-4}, 10^{-5}\}$). These results suggest that the regularization effect of hierarchical estimation, which is based on averaging over *more* samples (either virtual or actual), can positively impact the generalization of the learned models.[23]

Finally, we elaborate on how these results compare to the experiments on music annotation and retrieval (in Section 6.1). First, in the Character Trajectory Data Set the number of training sequences associated with each class (i.e, each character) is small compared to the CAL500 data set.[24] As a result, the EM-H3M algorithm is able to process all the data, and achieve good classification performance. However, EM-H3M still needs to evaluate the

23. For smaller values of ΔLL (e.g., $\Delta LL < 10^{-5}$), the performance of EM-H3M did not improve.

24. In the Character Trajectory data set there are on average 71 training sequences per character. In CAL500, each tag is associated with *thousands* of training sequences at the song level (e.g., an average of about 8000 audio fragments per tag).

	$N_v = 1000$				$\tau = 10$			
	$\tau = 2$	$\tau = 5$	$\tau = 10$	$\tau = 20$	$N_v = 1$	$N_v = 10$	$N_v = 100$	$N_v = 1000$
P@5	0.4656	0.4718	0.4738	0.4734	0.4775	0.4689	0.4689	0.4738
P@10	0.4437	0.4487	0.4507	0.4478	0.4534	0.4491	0.4466	0.4507
P@15	0.4236	0.4309	0.4346	0.4327	0.4313	0.4307	0.4242	0.4346

Table 6: Annotation and retrieval performance on CAL500 for VHEM-H3M when varying the virtual sample parameters N_v and τ.

likelihood of all the original sequences at each iteration. This leads to slower iterations, and results in a total training time about 5 times longer than that of VHEM-H3M (see Table 5). Second, the Character Trajectory data is more "controlled" than the CAL500 data, since each class corresponds to a single character, and all the examples are from the same writer. As a consequence, there is less variation in the intermediate HMMs (i.e., they are clustered more closely), and several of them may summarize the cluster well, providing good candidate cluster centers for PPK-SC. In conclusion, PPK-SC faces only a limited loss of information when selecting one of the initial HMMs to represent each cluster, and achieves reasonable performances.

6.3 Robustness of VHEM-H3M to Number and Length of Virtual Samples

The generation of virtual samples in VHEM-H3M is controlled by two parameters: the number of virtual sequences (N), and their length (τ). In this section, we investigate the impact of these parameters on annotation and retrieval performance on CAL500. For a given tag t, we set $N = N_v N_t K^{(s)}$, where N_v is a constant, N_t the number of training songs for the tag, and $K^{(s)}$ the number of mixture components for each song-level H3M. Starting with $(N_v, \tau) = (1000, 10)$, each parameter is varied while keeping the other one fixed, and annotation and retrieval performance on the CAL500 data set are calculated, as described in Section 6.1.

Table 6 presents the results, for $\tau \in \{2, 5, 10, 20\}$ and $N_v \in \{1, 10, 100, 1000\}$. The performances when varying τ are close on all metrics. For example, average P@5, P@10 and P@15 vary in small ranges (0.0082, 0.0070 and 0.0110, respectively). Similarly, varying the number of virtual sequences N_v has a limited impact on performance as well.[25] This demonstrates that VHEM-H3M is fairly robust to the choice of these parameters.

Finally, we tested VHEM-H3M for music annotation and retrieval on CAL500, using virtual sequences of the same length as the audio fragments used at the song level, that is, $\tau = T = 125$. Compared to $\tau = 10$ (the setting used in earlier experiments), we registered an 84% increase in total running time, with no corresponding improvement in performance. Thus, in our experimental setting, making the virtual sequences relatively short positively impacts the running time, without reducing the quality of the learned models.

25. Note that the E-step of the VHEM-H3M algorithm averages over all possible observations compatible with the input models, also when we choose a low value of N_v (e.g., $N_v = 1$). The number of virtual samples controls the "virtual mass" of each input HMMs and thus the certainty of cluster assignments.

7. Conclusion

In this paper, we presented a variational HEM (VHEM) algorithm for clustering HMMs with respect to their probability distributions, while generating a novel HMM center to represent each cluster. Experimental results demonstrate the efficacy of the algorithm on various clustering, annotation, and retrieval problems involving time-series data, showing improvement over current methods. In particular, using a two-stage, *hierarchical estimation* procedure—learn H3Ms on many smaller subsets of the data, and summarize them in a more compact H3M model of the data—the VHEM-H3M algorithm estimates annotation models from data more efficiently than standard EM and also improves model robustness through better regularization. Specifically, averaging over all possible virtual samples prevents over-fitting, which can improve the generalization of the learned models. Moreover, using relatively short virtual sequences positively impacts the running time of the algorithm, without negatively affecting its performance on practical applications. In addition, we have noted that the VHEM-H3M algorithm is robust to the choice of the number and length of virtual samples.

In our experiments, we have implemented the first stage of the hierarchical estimation procedure by partitioning data in *non-overlapping* subsets (and learning an intermediate H3M on each subset). In particular, partitioning the CAL500 data at the song level had a practical advantage. Since individual songs in CAL500 are relevant to several tags, the estimation of the song H3Ms can be executed one single time for each song in the database, and *re-used* in the VHEM estimation of all the associated tag models. This has a positive impact on computational efficiency. Depending on the particular application, however, a slightly different implementation of this first stage (of the hierarchical estimation procedure) may be better suited. For example, when estimating a H3M from a very large amount of training data, one could use a procedure that does not necessarily cover all data, inspired by Kleiner et al. (2011). If n is the size of the training data, first estimate $B > 1$ intermediate H3Ms on as many (possibly overlapping) "little" bootstrap subsamples of the data,[26] each of size $b < n$. Then summarize all the intermediate H3Ms into a final H3M using the VHEM-H3M algorithm.

In future work we plan to extend VHEM-H3M to the case where all HMMs share a large GMM universal background model for the emission distributions (with each HMM state having a different set of weights for the Gaussian components), which is commonly used in speech (Huang and Jack, 1989; Bellegarda and Nahamoo, 1990; Rabiner and Juang, 1993) or in hand-writing recognition (Rodriguez-Serrano and Perronnin, 2012). This would allow for faster training (moving the complexity to estimating the noise background model) and would require a faster implementation of the inference (e.g., using a strategy inspired by Coviello et al. (2012b)). In addition, we plan to derive a HEM algorithm for HMMs with discrete emission distributions, and compare its performance to the work presented here and to the extension with the large GMM background model.

Finally, in this work we have not addressed the model selection problem, that is, selecting the number of reduced mixture components. Since VHEM is based on maximum likelihood principles, it is possible to apply standard statistical model selection techniques, such as

26. Several techniques have been proposed to bootstrap from sequences of samples, for example refer to Hall et al. (1995).

Bayesian information criterion (BIC) and Akaike information criterion (AIC) (MacKay, 2003). Alternatively, inspired by Bayesian non-parametric statistics, the VHEM formulation could be extended to include a Dirichlet process prior (Blei and Jordan, 2006), with the number of components adapting to the data.

Acknowledgments

E.C. thanks Yingbo Song for providing the code for the PPK similarity between HMMs (Jebara et al., 2007) and assistance with the experiments on motion clustering. The authors acknowledge support from Google, Inc. E.C. and G.R.G.L. also wish to acknowledge support from Qualcomm, Inc., Yahoo!, Inc., KETI under the PHTM program, and NSF Grants CCF-0830535 and IIS-1054960. A.B.C. was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 110513). G.R.G.L. acknowledges support from the Alfred P. Sloan Foundation. This research was supported in part by the UCSD FWGrid Project, NSF Research Infrastructure Grant Number EIA-0303622.

Appendix A. Derivation of the Lower Bounds

The lower bounds are derived as follow.

A.1 Lower Bound on $\mathrm{E}_{\mathcal{M}_i^{(b)}}\left[\log p(Y_i|\mathcal{M}^{(r)})\right]$

The lower bound (15) on $\mathrm{E}_{\mathcal{M}_i^{(b)}}\left[\log p(Y_i|\mathcal{M}^{(r)})\right]$ is computed by introducing the variational distribution $q_i(z_i)$ and applying (10)

$$
\begin{aligned}
&\mathrm{E}_{\mathcal{M}_i^{(b)}}\left[\log p(Y_i|\mathcal{M}^{(r)})\right] \\
&\geq \max_{q_i} \sum_j q_i(z_i=j)\left\{\log\frac{p(z_i=j|\mathcal{M}^{(r)})}{q_i(z_i=j)} + \mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(Y_i|\mathcal{M}_j^{(r)})]\right\} \\
&= \max_{q_i} \sum_j q_i(z_i=j)\left\{\log\frac{p(z_i=j|\mathcal{M}^{(r)})}{q_i(z_i=j)} + \mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})^{N_i}]\right\} \qquad (30) \\
&= \max_{q_i} \sum_j q_i(z_i=j)\left\{\log\frac{p(z_i=j|\mathcal{M}^{(r)})}{q_i(z_i=j)} + N_i\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]\right\} \qquad (31) \\
&\geq \max_{q_i} \sum_j q_i(z_i=j)\left\{\log\frac{p(z_i=j|\mathcal{M}^{(r)})}{q_i(z_i=j)} + N_i\mathcal{L}_{HMM}^{i,j}\right\} \triangleq \mathcal{L}_{H3M}^{i}, \qquad (32)
\end{aligned}
$$

where in (30) we use the fact that Y_i is a set of N_i i.i.d. samples. In (31), $\log p(\mathbf{y}|\mathcal{M}_j^{(r)})$ is the observation log-likelihood of an HMM, which is essentially a mixture distribution. Since the latter expectation cannot be calculated directly, in (32) we use instead the lower bound $\mathcal{L}_{HMM}^{i,j}$ defined in (13).

A.2 Lower Bound on $\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]$

To calculate the lower bound $\mathcal{L}_{HMM}^{i,j}$,starting with (13), we first rewrite the expectation $\mathrm{E}_{\mathcal{M}_i^{(b)}}$ to explicitly marginalize over the HMM state sequence $\boldsymbol{\beta}$ from $\mathcal{M}_i^{(b)}$,

$$\begin{aligned}\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})] &= \mathrm{E}_{\boldsymbol{\beta}|\mathcal{M}_i^{(b)}}\left[\mathrm{E}_{\mathbf{y}|\boldsymbol{\beta},\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]\right] \\ &= \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \mathrm{E}_{\mathbf{y}|\boldsymbol{\beta},\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]. \end{aligned} \tag{33}$$

We introduce a variational distribution $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$ on the state sequence $\boldsymbol{\rho}$, which depends on a particular sequence $\boldsymbol{\beta}$ from $\mathcal{M}_i^{(b)}$. Applying (10) to (33), we have

$$\begin{aligned}&\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})] \\ &\geq \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \max_{q^{i,j}} \sum_{\boldsymbol{\rho}} q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{p(\boldsymbol{\rho}|\mathcal{M}_j^{(r)})}{q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \mathrm{E}_{\mathbf{y}|\boldsymbol{\beta},\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\boldsymbol{\rho},\mathcal{M}_j^{(r)})] \right\} \\ &= \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \max_{q^{i,j}} \sum_{\boldsymbol{\rho}} q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{p(\boldsymbol{\rho}|\mathcal{M}_j^{(r)})}{q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_t \mathrm{E}_{\mathcal{M}_{i,\beta_t}^{(b)}}[\log p(y_t|\mathcal{M}_{j,\rho_t}^{(r)})] \right\} \qquad (34) \\ &\geq \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \max_{q^{i,j}} \sum_{\boldsymbol{\rho}} q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{p(\boldsymbol{\rho}|\mathcal{M}_j^{(r)})}{q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_t \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \right\} \triangleq \mathcal{L}_{HMM}^{i,j}, \qquad (35)\end{aligned}$$

where in (34) we use the conditional independence of the observation sequence given the state sequence, and in (35) we use the lower bound, defined in (14), on each expectation.

A.3 Lower Bound on $\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})]$

To derive the lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ for (14), we first rewrite the expectation with respect to $\mathcal{M}_{i,\beta}^{(b)}$ to explicitly marginalize out the GMM hidden assignment variable ζ,

$$\begin{aligned}\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})] &= \mathrm{E}_{\zeta|\mathcal{M}_{i,\beta}^{(b)}}\left[\mathrm{E}_{\mathcal{M}_{i,\beta,\zeta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})]\right] \\ &= \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}\left[\log p(y|\mathcal{M}_{j,\rho}^{(r)})\right].\end{aligned}$$

Note that $p(y|\mathcal{M}_{j,\rho}^{(r)})$ is a GMM emission distribution as in (1). Hence, the observation variable is y, and the hidden variable is ζ. Therefore, we introduce the variational distribution $q_{\beta,\rho}^{i,j}(\zeta|m)$, which is conditioned on the observation y arising from the mth component in $\mathcal{M}_{i,\beta}^{(b)}$, and apply (10),

$$\begin{aligned}&\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})] \\ &\geq \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \max_{q_{\beta,\rho}^{i,j}} \sum_{\zeta=1}^{M} q_{\beta,\rho}^{i,j}(\zeta|m) \left\{ \log \frac{p(\zeta|\mathcal{M}_{j,\rho}^{(r)})}{q_{\beta,\rho}^{i,j}(\zeta|m)} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\zeta}^{(r)})] \right\} \triangleq \mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}.\end{aligned}$$

Appendix B. Derivation of the E-Step

GMM: Substituting the variational distribution $\eta_{\ell|m}^{(i,\beta),(j,\rho)}$ into (17), we have

$$\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)} = \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \max_{\eta_{\ell|m}^{(i,\beta),(j,\rho)}} \sum_{\ell=1}^{M} \eta_{\ell|m}^{(i,\beta),(j,\rho)} \left\{ \log \frac{c_{\rho,\ell}^{(r),j}}{\eta_{\ell|m}^{(i,\beta),(j,\rho)}} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right\}.$$

The maximizing variational parameters are obtained as (see Appendix D.2)

$$\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)} = \frac{c_{\rho,\ell}^{(r),j} \exp\left\{ \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right\}}{\sum_{\ell'} c_{\rho,\ell'}^{(r),j} \exp\left\{ \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell'}^{(r)})] \right\}},$$

HMM: Substituting the variational distribution $\phi^{i,j}$ into (16), we have

$$\mathcal{L}_{HMM}^{i,j} = \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \max_{\phi^{i,j}} \sum_{\boldsymbol{\rho}} \phi^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{\pi_{\boldsymbol{\rho}}^{(r),j}}{\phi^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_{t} \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \right\}. \tag{36}$$

The maximization of (36) with respect to $\phi_t^{i,j}(\rho_t|\rho_{t-1},\beta_t)$ and $\phi_1^{i,j}(\rho_1|\beta_1)$ is carried out independently for each pair (i,j), and follow Hershey et al. (2007). In particular it uses a backward recursion, starting with $\mathcal{L}_{\tau+1}^{i,j}(\beta_t,\rho_t) = 0$, for $t = \tau,\ldots,2$,

$$\hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) = \frac{a_{\rho_{t-1},\rho_t}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta_t),(j,\rho_t)} + \mathcal{L}_{t+1}^{i,j}(\beta_t,\rho_t) \right\}}{\sum_{\rho}^{S} a_{\rho_{t-1},\rho}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta_t),(j,\rho)} + \mathcal{L}_{t+1}^{i,j}(\beta_t,\rho) \right\}}$$

$$\mathcal{L}_t^{i,j}(\beta_{t-1},\rho_{t-1}) = \sum_{\beta=1}^{S} a_{\beta_{t-1},\beta}^{(b),i} \log \sum_{\rho=1}^{S} a_{\rho_{t-1},\rho}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta),(j,\rho)} + \mathcal{L}_{t+1}^{i,j}(\beta,\rho) \right\},$$

and terminates with

$$\hat{\phi}_1^{i,j}(\rho_1|\beta_1) = \frac{\pi_{\rho_1}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta_1),(j,\rho_1)} + \mathcal{L}_2^{i,j}(\beta_1,\rho_1) \right\}}{\sum_{\rho}^{S} \pi_{\rho}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta_1),(j,\rho)} + \mathcal{L}_2^{i,j}(\beta_1,\rho) \right\}}$$

$$\mathcal{L}_{\mathrm{HMM}}^{i,j} = \sum_{\beta=1}^{S} \pi_{\beta}^{(b),i} \log \sum_{\rho=1}^{S} \pi_{\rho}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta),(j,\rho)} + \mathcal{L}_2^{i,j}(\beta,\rho) \right\} \tag{37}$$

where (37) is the maxima of the terms in (36) in Section 3.3.1.
H3M: Substituting the variational distribution z_{ij} into (15), we have

$$\mathcal{L}_{H3M}^{i} = \max_{z_{ij}} \sum_{j} z_{ij} \left\{ \log \frac{\omega_j^{(r)}}{z_{ij}} + N_i \mathcal{L}_{HMM}^{i,j} \right\}. \tag{38}$$

The maximizing variational parameters of (38) are obtained using Appendix D.2,

$$\hat{z}_{ij} = \frac{\omega_j^{(r)} \exp(N_i \mathcal{L}_{HMM}^{i,j})}{\sum_{j'} \omega_{j'}^{(r)} \exp(N_i \mathcal{L}_{HMM}^{i,j'})}.$$

Appendix C. Derivation of the M-Step

The M-steps involves maximizing the lower bound in (11) with respect to $\mathcal{M}^{(r)}$, while holding the variational distributions fixed,

$$\mathcal{M}^{(r)^*} = \operatorname*{argmax}_{\mathcal{M}^{(r)}} \sum_{i=1}^{K^{(b)}} \mathcal{L}^i_{H3M}. \tag{39}$$

Substituting (20) and (21) into the objective function of (39),

$$\begin{aligned}\mathcal{L}(\mathcal{M}^{(r)}) &= \sum_{i=1}^{K^{(b)}} \mathcal{L}^i_{H3M} \\ &= \sum_{i,j} \hat{z}_{ij} \left\{ \log \frac{\omega_j^{(r)}}{\hat{z}_{ij}} + N_i \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \sum_{\boldsymbol{\rho}} \hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left[\log \frac{\pi_{\boldsymbol{\rho}}^{(r),j}}{\hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_t \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \right] \right\}\end{aligned} \tag{40}$$

In the following, we detail the update rules for the parameters of the reduced model $\mathcal{M}^{(r)}$.

C.1 HMMs Mixture Weights

Collecting terms in (40) that only depend on the mixture weights $\{\omega_j^{(r)}\}_{j=1}^{K^{(r)}}$, we have

$$\widetilde{\mathcal{L}}(\{\omega_j^{(r)}\}) = \sum_i \sum_j \hat{z}_{ij} \log \omega_j^{(r)} = \sum_j \left[\sum_i \hat{z}_{ij} \right] \log \omega_j^{(r)} \tag{41}$$

Given the constraints $\sum_{j=1}^{K^{(r)}} \omega_j^{(r)} = 1$ and $\omega_j^{(r)} \geq 0$, (41) is maximized using the result in Appendix D.1, which yields the update in (24).

C.2 Initial State Probabilities

The objective function in (40) factorizes for each HMM $\mathcal{M}_j^{(r)}$, and hence the parameters of each HMM are updated independently. For the j-th HMM, we collect terms in (40) that depend on the initial state probabilities $\{\pi_\rho^{(r),j}\}_{\rho=1}^{S}$,

$$\begin{aligned}\widetilde{\mathcal{L}}_j(\{\pi_\rho^{(r),j}\}) &= \sum_i \hat{z}_{ij} N_i \sum_{\beta_1} \pi_{\beta_1}^{(b),i} \sum_{\rho_1} \hat{\phi}_1^{i,j}(\rho_1|\beta_1) \log \pi_{\rho_1}^{(r),j} \\ &= \sum_{\rho_1} \sum_i \hat{z}_{ij} N_i \underbrace{\sum_{\beta_1} \pi_{\beta_1}^{(b),i} \hat{\phi}_1^{i,j}(\rho_1|\beta_1)}_{\hat{\nu}_1^{i,j}(\rho_1)} \log \pi_{\rho_1}^{(r),j} \\ &= \sum_\rho \sum_i \hat{z}_{ij} N_i \hat{\nu}_1^{i,j}(\rho) \log \pi_\rho^{(r),j} \qquad (42) \\ &\propto \sum_\rho \left[\sum_i \hat{z}_{ij} \omega_i^{(b)} \hat{\nu}_1^{i,j}(\rho) \right] \log \pi_\rho^{(r),j}, \qquad (43)\end{aligned}$$

where in the (42) we have used the summary statistic defined in (23). Considering the constraints $\sum_{\rho=1}^{S} \pi_{\rho}^{(r),j} = 1$ and $\pi_{\rho}^{(r),j} \geq 0$, (43) is maximized using the result in Appendix D.1, giving the update formula in (25).

C.3 State Transition Probabilities

Similarly, for each HMM $\mathcal{M}_j^{(r)}$ and previous state ρ, we collect terms in (40) that depend on the transition probabilities $\{a_{\rho,\rho'}^{(r),j}\}_{\rho'=1}^{S}$,

$$
\begin{aligned}
\widetilde{\mathcal{L}}_{j,\rho}(\{a_{\rho,\rho'}^{(r),j}\}_{\rho'=1}^{S}) &= \sum_i \hat{z}_{ij} N_i \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \sum_{\boldsymbol{\rho}} \hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \log \pi_{\boldsymbol{\rho}}^{(r),j} \\
&\propto \sum_i \hat{z}_{ij} N_i \sum_{\boldsymbol{\beta}} \left[\pi_{\beta_1}^{(b),i} \prod_{t=2}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \right] \sum_{\boldsymbol{\rho}} \left[\hat{\phi}_1^{i,j}(\rho_1|\beta_1) \prod_{t=2}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \right] \left[\sum_{t=2}^{\tau} \log a_{\rho_{t-1},\rho_t}^{(r),j} \right] \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\rho_1} \sum_{\rho_2} \sum_{\beta_2} \underbrace{\sum_{\beta_1} \underbrace{\pi_{\beta_1}^{(b),i} \hat{\phi}_1^{i,j}(\rho_1|\beta_1)}_{\nu_1^{i,j}(\rho_1,\beta_1)} a_{\beta_1,\beta_2}^{(b),i} \hat{\phi}_2^{i,j}(\rho_2|\rho_1,\beta_2)}_{\xi_2^{i,j}(\rho_1,\rho_2,\beta_2)} \Big[\log a_{\rho_1,\rho_2}^{(r),j} \\
&\quad + \sum_{\beta_3 \cdots \beta_\tau} \sum_{\rho_3 \cdots \rho_\tau} \prod_{t=3}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \prod_{t=3}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \sum_{t=3}^{\tau} \log a_{\rho_{t-1},\rho_t}^{(r),j} \Big] \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\rho_1} \sum_{\rho_2} \sum_{\beta_2} \xi_2^{i,j}(\rho_1,\rho_2,\beta_2) \log a_{\rho_1,\rho_2}^{(r),j} \\
&\quad + \sum_i \hat{z}_{ij} N_i \sum_{\rho_2} \sum_{\rho_3} \sum_{\beta_3} \underbrace{\sum_{\beta_2} \underbrace{\sum_{\rho_1} \xi_2^{i,j}(\rho_1,\rho_2,\beta_2)}_{\nu_2^{i,j}(\rho_2,\beta_2)} a_{\beta_2,\beta_3}^{(b),i} \hat{\phi}_3^{i,j}(\rho_3|\rho_2,\beta_3)}_{\xi_3^{i,j}(\rho_2,\rho_3,\beta_3)} \Big[\log a_{\rho_2,\rho_3}^{(r),j} \\
&\quad + \sum_{\beta_4 \cdots \beta_\tau} \sum_{\rho_4 \cdots \rho_\tau} \prod_{t=4}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \prod_{t=4}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \sum_{t=4}^{\tau} \log a_{\rho_{t-1},\rho_t}^{(r),j} \Big] \\
&= \dots \\
&= \sum_i \hat{z}_{ij} N_i \sum_{t=2}^{\tau} \sum_{\rho_{t-1}} \sum_{\rho_t} \sum_{\beta_t} \xi_t^{i,j}(\rho_{t-1},\rho_t,\beta_t) \log a_{\rho_{t-1},\rho_t}^{(r),j} \\
&\propto \sum_i \hat{z}_{ij} N_i \sum_{\rho'} \underbrace{\sum_{t=2}^{\tau} \sum_{\beta} \xi_t^{i,j}(\rho,\rho',\beta)}_{\hat{\xi}^{i,j}(\rho,\rho')} \log a_{\rho,\rho'}^{(r),j} \\
&\propto \sum_{\rho'=1}^{S} \left[\sum_i \hat{z}_{ij} \omega_i^{(b)} \hat{\xi}^{i,j}(\rho,\rho') \right] \log a_{\rho,\rho'}^{(r),j}. \qquad (44)
\end{aligned}
$$

Considering the constraints $\sum_{\rho'=1}^{S} a_{\rho,\rho'}^{(r),j} = 1$ and $a_{\rho,\rho'}^{(r),j} \geq 0$, (44) is maximized using the result in Appendix D.1, giving the update in (25).

C.4 Emission Probability Density Functions

The cost function (40) factors also for each GMM indexed by (j, ρ, ℓ). Factoring (40),

$$
\begin{aligned}
\widetilde{\mathcal{L}}(\mathcal{M}_{j,\rho,\ell}^{(r)}) &= \sum_i \hat{z}_{ij} N_i \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \sum_{\boldsymbol{\rho}} \hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \sum_t \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\boldsymbol{\beta}} \left[\pi_{\beta_1}^{(b),i} \prod_{t=2}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \right] \sum_{\boldsymbol{\rho}} \left[\hat{\phi}_1^{i,j}(\rho_1|\beta_1) \prod_{t=2}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \right] \sum_t \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\rho_1} \sum_{\beta_1} \underbrace{\pi_{\beta_1}^{(b),i} \hat{\phi}_1^{i,j}(\rho_1|\beta_1)}_{\nu_1^{i,j}(\rho_1,\beta_1)} \Big[\mathcal{L}_{GMM}^{(i,\beta_1),(j,\rho_1)} \dots \\
&\qquad + \sum_{\beta_2 \cdots \beta_\tau} \prod_{t=2}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \sum_{\rho_2 \cdots \rho_\tau} \prod_{t=2}^{\tau} \hat{\phi}^{i,j}(\rho_t|\rho_{t-1},\beta_t) \sum_{t=2}^{\tau} \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \Big] \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\rho_1} \sum_{\beta_1} \nu_1^{i,j}(\rho_1,\beta_1) \mathcal{L}_{GMM}^{(i,\beta_1),(j,\rho_1)} \\
&\quad + \sum_i \hat{z}_{ij} N_i \sum_{\rho_2} \sum_{\beta_2} \underbrace{\sum_{\rho_1} \underbrace{\sum_{\beta_1} \left(\nu_1^{i,j}(\rho_1,\beta_1) a_{\beta_1,\beta_2}^{(b),i} \right) \hat{\phi}_2^{i,j}(\rho_2|\rho_1,\beta_2)}_{\xi_2^{i,j}(\rho_1,\rho_2,\beta_2)}}_{\nu_2^{i,j}(\rho_2,\beta_2)} \Big[\mathcal{L}_{GMM}^{(i,\beta_2),(j,\rho_2)} \dots \\
&\qquad + \sum_{\beta_3 \cdots \beta_\tau} \prod_{t=3}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \sum_{\rho_3 \cdots \rho_\tau} \prod_{t=3}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \sum_{t=3}^{\tau} \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \Big] \\
&= \dots \\
&= \sum_i \hat{z}_{ij} N_i \sum_{t=1}^{\tau} \sum_{\rho_t} \sum_{\beta_t} \nu_t^{i,j}(\rho_t,\beta_t) \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \\
&\propto \sum_i \hat{z}_{ij} N_i \sum_{\beta} \underbrace{\sum_{t=1}^{\tau} \nu_t^{i,j}(\rho,\beta)}_{\hat{\nu}_t^{i,j}(\rho,\beta)} \mathcal{L}_{GMM}^{(i,\beta),(j,\rho)} \\
&\propto \sum_i \hat{z}_{ij} N_i \sum_{\beta=1}^{S} \hat{\nu}^{i,j}(\rho,\beta) \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \, \hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)} \left[\log c_{\rho,\ell}^{(r),j} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right] \\
&= \Omega_{j,\rho} \left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)} \left[\log c_{\rho,\ell}^{(r),j} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right] \right), \qquad (45)
\end{aligned}
$$

where in (45) we use the weighted-sum operator defined in (28), which is over all base model GMMs $\{\mathcal{M}_{i,\beta,m}^{(b)}\}$. The GMM mixture weights are subject to the constraints $\sum_{\ell=1}^{M} c_{\rho,\ell}^{(r),j} = 1$,

$\forall j, \rho$. Taking the derivative with respect to each parameter and setting it to zero,[27] gives the GMM update Equations (26) and (27).

Appendix D. Useful Optimization Problems

The following two optimization problems are used in the derivation of the E-step and M-step.

D.1

The optimization problem

$$\max_{\alpha_\ell} \quad \sum_{\ell=1}^{L} \beta_\ell \log \alpha_\ell \qquad \text{s.t.} \qquad \sum_{\ell=1}^{L} \alpha_\ell = 1, \quad \alpha_\ell \geq 0, \forall \ell$$

is optimized by $\alpha_\ell^* = \frac{\beta_\ell}{\sum_{\ell'=1}^{L} \beta'_\ell}$.

D.2

The optimization problem

$$\max_{\alpha_\ell} \quad \sum_{\ell=1}^{L} \alpha_\ell \left(\beta_\ell - \log \alpha_\ell\right) \qquad \text{s.t.} \qquad \sum_{\ell=1}^{L} \alpha_\ell = 1 \quad \alpha_\ell \geq 0, \forall \ell$$

is optimized by $\alpha_\ell^* = \frac{\exp \beta_\ell}{\sum_{\ell'=1}^{L} \exp \beta'_\ell}$.

Appendix E. Clustering Synthetic Data where the Clustering Model Does Not Match the "True" Model

In this appendix we present two experiments on clustering synthetic data, where the order of the model used for clustering does not necessarily match the order of the "true" model used to generate the data. In both experiments, the true model consists of $C = 4$ HMMs each with $S = 3$ hidden states—we used the HMMs of experiment (c) in Section 5.3. Data sequences are generated from each of the C HMMs, and then corrupted with Gaussian noise. An HMM with S' states is estimated over each sequence. These HMMs are denoted as *noisy* HMMs, to differentiate them from the original HMMs representing the C classes. The entirety of noisy HMMs are then clustered into C' groups using in turn our VHEM-H3M algorithm and PPK-SC.

In the first experiments we set $S' = S(= 3)$, and vary $C' \in \{3, 4, 5, 6\}$. Hence, the number of clusters does not necessarily match the true number of groups. In the second experiments, we fix $C' = C(= 4)$, and vary $S' \in \{2, 3, 4, 5\}$, that is, the model order of the noisy HMMs does not match the order of the original HMMs. Results are reported in Table 7 below.

27. We also considered the constraints on the covariance matrices $\Sigma_{\rho,\ell}^{(r),j} \succ \mathbf{0}$.

	experiment (d) number of clusters C' varies, $S' = 3$				experiment (e) number of HMM states S' varies, $C' = 4$ fixed			
	2	3	4 (true)	5	2	3 (true)	4	5
VHEM-H3M	0.665	0.782	0.811	0.816	0.801	0.811	0.828	0.832
PPK-SC	0.673	0.729	0.768	0.781	0.766	0.768	0.781	0.798

Table 7: Results on clustering synthetic data with VHEM-H3M and PPK-SC, when the model order does not match the order of the true model used for generating the data. We fist vary the number of clusters C', keeping $S' = 3$ fixed to the true value. Then, we vary the number of HMM states S', keeping $C' = 4$ fixed to the true value. Performance is measured in terms of Rand-index, and is averaged over $K \in \{2, 4, 8, 16, 32\}$.

Performance are more sensitive to selecting a sufficient number of clusters than using the right number of HMM states. In particular, when using fewer clusters than the true number of classes (e.g., $C' < C$), the Rand-index degrades for both VHEM-H3M and PPK-SC, see experiment (d) in Table 7. On the opposite, performance are relatively stable when the number of HMM states does not match the true one, for example, $S' \neq S$, see experiment (e) in Table 7. In particular, when using fewer HMM states (e.g., $S' < S$) the model can still capture some of the dynamics, and the drop in performance is not significant. It is also interesting to note that using a larger number of HMM states (e.g., $S' > S$) leads to slightly better results. The reason is that, when estimating the HMMs on the corrupted data sequences, there are additional states to better account for the effect of the noise.

References

Dimitris Achlioptas, Frank McSherry, and Bernhard Schölkopf. Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems 14*, volume 1, pages 335–342. MIT Press, 2002.

Jonathan Alon, Stan Sclaroff, George Kollios, and Vladimir Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 368–375. IEEE, 2003.

Arthur Asuncion and David J Newman. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

Claus Bahlmann and Hans Burkhardt. Measuring HMM similarity with the Bayes probability of error and its application to online handwriting recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 406–411. IEEE, 2001.

Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

Eloi Batlle, Jaume Masip, and Enric Guaus. Automatic song identification in noisy broadcast audio. In *Proceeding of the International Conference on Signal and Image Processing*. IASTED, 2002.

Jerome R Bellegarda and David Nahamoo. Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(12):2033–2045, 1990.

Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems*, volume 16, pages 177–184. MIT Press, 2004.

David M Blei and Michael I Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006.

William M Campbell. A SVM/HMM system for speaker recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–209. IEEE, 2003.

Gustavo Carneiro, Antoni B Chan, Pedro J Moreno, and Nuno Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):394–410, 2007.

Antoni B Chan, Emanuele Coviello, and Gert RG Lanckriet. Clustering Dynamic Textures with the Hierarchical EM Algorithm. In *Proceeding of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010a.

Antoni B Chan, Emanuele Coviello, and Gert RG Lanckriet. Derivation of the hierarchical EM Algorithm for Dynamic Textures. Technical report, City University of Hong Kong, 2010b.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning sequence kernels. In *IEEE Workshop on Machine Learning for Signal Processing*, pages 2–8. IEEE, 2008.

Emanuele Coviello, Antoni B Chan, and Gert RG Lanckriet. Time series models for semantic music annotation. *IEEE Transactions on Audio, Speech, and Language Processing*, 5(19): 1343–1359, 2011.

Emanuele Coviello, Antoni Chan, and Gert Lanckriet. The variational hierarchical EM algorithm for clustering hidden Markov models. In *Advances in Neural Information Processing Systems 25*, pages 413–421, 2012a.

Emanuele Coviello, Adeel Mumtaz, Antoni B Chan, and Gert RG Lanckriet. Growing a Bag of Systems Tree for fast and accurate classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1979–1986. IEEE, 2012b.

Imre Csiszár and Gábor Tusnády. Information geometry and alternating minimization procedures. *Statistics and decisions*, 1984.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

Inderjit S. Dhillon. Differential entropic clustering of multivariate Gaussians. In *Advances in Neural Information Processing Systems*, volume 19, page 337. MIT Press, 2007.

Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal on Computer Vision*, 51(2):91–109, 2003.

Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, pages 385–392. MIT Press, 2008.

Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

Jacob Goldberger and Sam Roweis. Hierarchical clustering of a mixture model. In *Advances in Neural Information Processing Systems*, pages 505–512. MIT Press, 2004.

Asela Gunawardana and William Byrne. Convergence theorems for generalized alternating minimization procedures. *Journal of Machine Learning Research*, 6:2049–73, 2005.

Peter Hall, Joel L Horowitz, and Bing-Yi Jing. On blocking rules for the bootstrap with dependent data. *Biometrika*, 82(3):561–574, 1995.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

Christian Hennig. Cluster-wise assessment of cluster stability. *Computational Statistics & Data Analysis*, 52(1):258–271, 2007.

John R Hershey and Peder A Olsen. Variational Bhattacharyya divergence for hidden Markov models. In *IEEE Computer Society International Conference on Acoustics, Speech and Signal Processing*, pages 4557–4560. IEEE, 2008.

John R Hershey, Peder A Olsen, and Steven J Rennie. Variational Kullback-Leibler divergence for hidden Markov models. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 323–328. IEEE, 2007.

Matthew D Hoffman, David M Blei, and Perry R Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of the 10th International Symposium on Music Information Retrieval*, pages 369–374, 2009.

Xuedong D Huang and Mervyn A Jack. Semi-continuous hidden Markov models for speech signals. *Computer Speech & Language*, 3(3):239–251, 1989.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2 (1):193–218, 1985.

Tommi S. Jaakkola. Tutorial on Variational Approximation Methods. In *Advanced Mean Field Methods: Theory and Practice*, pages 129–159. MIT Press, 2000.

Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004.

Tony Jebara, Yingbo Song, and Kapil Thadani. Spectral clustering and embedding with hidden Markov models. In *Machine Learning: ECML 2007*, pages 164–175. 2007.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Biing-Hwang Juang and Lawrence R Rabiner. A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal*, 64(2):391–408, February 1985.

Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L1-Norm and Related Methods*, pages 405–416, 1987.

Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

Eamonn J Keogh and Michael J Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 285–289. ACM, 2000.

Ariel Kleiner, Ameet Talwalkar, Purnamrita Sarkar, and Michael I Jordan. Bootstrapping big data. *Advances in Neural Information Processing Systems, Workshop: Big Learning: Algorithms, Systems, and Tools for Learning at Scale*, 2011.

Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. Hidden Markov models in computational biology. Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, 1994.

Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Scalable algorithms for string kernels with inexact matching. In *Advances in Neural Information Processing Systems*, pages 881–888, 2008.

Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 7, pages 566–575, 2002.

Marcus Liwicki and Horst Bunke. Iam-ondb-an on-line english sentence database acquired from handwritten text on a whiteboard. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 956–961. IEEE, 2005.

Rune B Lyngso, Christian NS Pedersen, and Henrik Nielsen. Metrics and similarity measures for hidden Markov models. In *Proceedings International Conference on Intelligent Systems for Molecular Biology*, pages 178–186, 1999.

David JC MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge university press, 2003.

Michael I Mandel and Daniel PW Ellis. Multiple-instance learning for music information retrieval. In *Proceedings of the 9th International Symposium on Music Information Retrieval*, pages 577–582, 2008.

Nag, Kin-Hong Wong, and Frank Fallside. Script recognition using hidden Markov models. In *IEEE Computer Society International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 2071–2074. IEEE, 1986.

Radford M Neal and Geoffrey E Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, 89:355–370, 1998.

Evert J Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930.

Tim Oates, Laura Firoiu, and Paul R Cohen. Clustering time series with hidden markov models and dynamic time warping. In *Joint Conferences on Artificial Intelligence, Workshop on Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning*, pages 17–21, 1999.

Antonello Panuccio, Manuele Bicego, and Vittorio Murino. A hidden Markov model-based approach to sequential data clustering. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 734–743, 2002.

William D Penny and Stephen J Roberts. Notes on variational learning. Technical report, Oxford University, 2000.

Yuting Qi, John William Paisley, and Lawrence Carin. Music analysis using hidden Markov mixture models. *IEEE Transactions on Signal Processing*, 55(11):5209–5224, 2007.

Lawrence R Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Upper Saddle River (NJ, USA), 1993.

Jeremy Reed and Chin-Hui Lee. A study on music genre classification based on universal acoustic models. In *Proceedings of the 7th International Symposium on Music Information Retrieval*, pages 89–94, 2006.

Jose Rodriguez-Serrano and Florent Perronnin. A model-based sequence similarity with application to handwritten word-spotting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:2108 – 2120, 2012.

Nicolas Scaringella and Giorgio Zoia. On the modeling of time information for automatic genre recognition systems in audio signals. In *Proc. 6th International Symposium Music Information Retrieval*, pages 666–671, 2005.

Padhraic Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems*, pages 648–654. MIT Press, 1997.

Theodoros Theodoridis and Huosheng Hu. Action classification of 3d human models using dynamic ANNs for mobile robot surveillance. In *IEEE Computer Society International Conference on Robotics and Biomimetics*, pages 371–376. IEEE, 2007.

Douglas Turnbull, Luke Barrington, David Torres, and Gert RG Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.

Nuno Vasconcelos. Image indexing with mixture hierarchies. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

Nuno Vasconcelos and Andrew Lippman. Learning mixture hierarchies. In *Advances in Neural Information Processing Systems*, 1998.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Ben H Williams, Marc Toussaint, and Amos J Storkey. Extracting motion primitives from natural handwriting data. In *Proceeding of the 16th International Conference on Artificial Neural Networks*, pages 634–643. Springer, 2006.

Jie Yin and Qiang Yang. Integrating hidden Markov models and spectral analysis for sensory time series clustering. In *IEEE Computer Society International Conference on Data Mining*. IEEE, 2005.

Shi Zhong and Joydeep Ghosh. A unified framework for model-based clustering. *The Journal of Machine Learning Research*, 4:1001–1037, 2003.