FlowHMM: Flow-based continuous hidden Markov models

Anonymous Authors[1]

Abstract

Continuous hidden Markov models (HMMs) assume that observations are generated from a mixture of Gaussian densities, limiting their ability to model more complex distributions. In this work, we address this shortcoming and propose a novel continuous HMM that allows to learn general continuous observation densities without constraining them to follow a Gaussian distribution or their mixtures. To that end, we leverage deep flow-based architectures that model complex, non-Gaussian functions. Moreover, to simplify optimization and avoid costly expectation-maximization algorithm, we use the co-occurrence matrix of discretized observations and consider the joint distribution of pairs of co-observed values. Even though our model is trained on discretized observations, it represents a continuous variant of HMM during inference, thanks to applying a separate flow model for each hidden state. The experiments on synthetic and real datasets show that our method outperforms Gaussian baselines.

1. Introduction

Hidden Markov models (HMMs) are a standard tool in modeling and analysis of time series data. Although structurally simple, they have been successfully applied in a wide variety of applications, ranging from finance (Mamon & Elliott, 2007), speech recognition (Gales & Young, 2008) to computational biology (Vidyasagar, 2014) and climate modeling (Ailliot et al., 2009).

HMMs are capable of solving complex problems, but their adoption is limited due to several reasons. First, the training process of HMMs typically relies on the Baum-Welch algorithm (Baum et al., 1970), a particular case of expectation-maximization (EM) method, which offers a relatively slow convergence to a local maximum. To reduce this burden, several discrete HHMs introduce the so-called co-occurrence matrix that aggregates information about the probability of jointly observed values in the chain and estimate it together with the parameters of the whole model (Huang et al., 2018; Lakshminarayanan & Raich, 2010; Hsu et al., 2012; Sicking et al., 2020; Mattila et al., 2017). Although the co-occurrence matrix is computed only once and then used to significantly reduce convergence time during optimization, its application is strictly limited to discrete HMMs and cannot be easily generalized to their continuous variants.

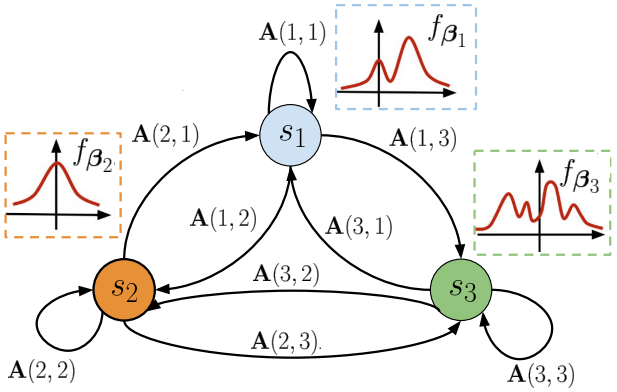

Figure 1: The concept of FlowHMM for $L = 3$ states and transition matrix $\mathbf{A}$. Each emission distribution characterised by density $f_{\boldsymbol{\beta}_l(\cdot)}$ is modeled using a separate flow component. Thanks to this, they can adjust to complex, non-Gaussian distributions.

Secondly, continuous HMM models are restricted to follow standard parametrized distributions when modeling the observations. Most of them use either Gaussians or other parametric families of distributions (Kontorovich et al., 2013; Darwin & Kiefer, 2020), while the others rely on the mixtures of Gaussians or apply semiparametric distribution estimation (Dannemann, 2012; Yu, 2018). As a result, the existing HMMs have limited ability to model complex observations that do not follow the distributions mentioned above. This, in turn, hinders their application in real-life use cases, *e.g.* in human action recognition (Xia et al., 2012). In this work, we address the shortcomings of the existing HMMs and introduce FlowHMM, a novel continuous hidden Markov model that learns a general continuous

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

distribution of observations by exploiting the properties of flow-based models (Rezende & Mohamed, 2015). The core idea of our approach is to model the distributions of the observations with normalizing flows, instead of Gaussians or their mixtures. Flows map a simple Gaussian prior to more complex distributions using a parametrized neural network. This formulation enables flow-based models to overpass parametric models in their flexibility to handle data samples of unknown distributions. Moreover, flow-based models naturally extend to a multimodal setting, which effectively renders obsolete the tedious process of tuning the number of hidden states.

Practically, training the model with flow-based components requires a gradient-based optimization. To achieve that, we discretize the continuous values during training and leverage the co-occurrence matrix. As a result, we provide an end-to-end training procedure that jointly optimizes the flow-based component parameters and the co-occurrence matrix using standard gradient-based techniques. Since the model is trained in a discretized form, the optimization process is simple and more efficient than the competing HMM training procedures, while during inference, we can still use a continuous version of our FlowHMM.

To summarize, the *main contribution of our paper* is a novel continuous HMM method dubbed FlowHMM, capable of modeling complex multimodal distributions of observations without constraining them to follow Gaussians. Not only does it outperform the competing approaches, but it also increases the efficiency of the required optimization procedure.

2. Related work

Second and higher order learning of parameters for discrete HMMs. One of (several) drawbacks of using Baum-Welch algorithm for training is that it is prohibitively slow for long sequences. The complexity of the forward-backward algorithm is $\mathcal{O}(N^2T)$, where N is the number of hidden states and T is the length of the observation sequence. The forward-backward algorithm needs to be called at each iteration of the Baum-Welch algorithm.

On the other hand, for long sequences, it turns out that it is enough to work with some statistics of observations. When T is large, we may accurately estimate the co-occurrence probability between two consecutive observations $\mathbb{P}(y_k, y_{k+1})$. We thus estimate these probabilities (collecting them in the so-called co-occurrence matrix) and optimize the model in such a way that the estimated probabilities are close (in some sense) to probabilities yielded by the model. One may also consider *e.g.*, triplets $\mathbb{P}(y_k, y_{k+1}, y_{k+2})$ or higher dimensional tuples. The identifiability of HMMs from pairwise co-occurrence probabilities is studied in (Huang et al., 2018) (higher order statistics are also discussed there) – authors *prove* that it is possible to recover the structure of an HMM based only on co-occurrences. An interesting extension (both, of pairs and triplets) was considered in (Mattila et al., 2020). Authors used there non-consecutive (*e.g.*, pairs $(y_k, y_{k+\tau})$ with some $\tau > 1$) tuples, which outperforms consecutive tuples (*i.e.*, $\tau = 1$) in some cases.

Our approach is, in a sense, close to the technique used in (Lakshminarayanan & Raich, 2010), with however significant differences: authors use alternating least square methods for optimization (in our case, all the matrices are real-valued and, softmax is applied whenever needed) and for the continuous case, they assume that the observation densities are a mixture of the predefined number of kernels. The "softmax trick" is also used in (Sicking et al., 2020) (their model learns so-called *dense* representations of hidden states and observation probabilities – it is designed only for discrete HMMs).

Continuous HMMs. As already mentioned, typically continuous HMMs assume that observations were sampled from a Gaussian distribution or a mixture of such distributions. For a non-Gaussian distribution, one usually assumes some parametric family of distributions. In (Kontorovich et al., 2013) authors propose a *decoupling* method to learning the parametric HMMs which are stationary. Instead of estimating the parameters of hidden states and observations jointly, they *i)* learn the parameters of observation densities (using some parametric mixture learner), and then *ii)* hidden states probabilities are learned by solving some convex quadratic programming. In (Dannemann, 2012) a method is proposed for a case where at least one state-dependent distribution is modeled with some nonparametric technique (*e.g.*, maximum likelihood estimation under shape constraints), some broader review of methods is presented in (Dannemann et al., 2014). In (Yu, 2018) authors use a Gaussian copula to model the dependence structure. A semiparametric data transformation is also proposed to ensure one may indeed use such copulas (the final observation distribution is a finite mixture of the copula models). In (Liu et al., 2015) authors present an efficient learning for parametric continuous HMMs sampled at finite irregular time instants (they incorporate some ideas from the theory of continuous-time Markov chains).

Number of hidden states. Another problem with HMMs is choosing the "*correct*" number of hidden states. The state-of-the-art approaches usually train multiple HMMs with different number of states, and the final one is selected according to some predefined criteria (*e.g.*, the Akaike information criterion (AIC) (Cavanaugh & Neath, 2019) or the Bayesian Information Criterion (BIC) (Schwarz, 1978)).

The criteria take into account not only the log-likelihood of observations but also the number of parameters of the whole model (in order not to overfit it). A HDP-HMM (hierarchical Dirichlet HMM) (Fox et al., 2008) is a non-parametric model which learns the number of hidden states automatically, but the method is computationally intensive in case of a very large datasets (Du et al., 2010). Though our method does not find the optimal number of hidden states automatically, setting it to *correct* value is not crucial since it leverages the ability of flow-based models to learn multimodal distributions.

Neural networks and HMMs. A link between HMMs and Multilayer Perceptrons was already pointed out in (Bourlard & Wellekens, 1990), a model which is trained by gradient methods was later presented in (Salazar et al., 2003). Incorporating a simple feed-forward network, convolutional neural network and stacked LSTMs in HMMs is presented in (Tran et al., 2016). In (Ilhan et al., 2020) a recurrent neural network (RNN) employed HMM for transitions – each regime controls hidden state transitions of the recurrent cell independently.

In (Ghosh et al., 2020) authors employ normalizing flows in observation densities, their model is trained using EM algorithm. They study classification problems using such a model. In (de Bézenac et al., 2020), the authors consider the extension of a Kalman filter (which you can think of as a version of a HMM with continuous both, observations and hidden states), where the *pseudo-observations* (as authors call it) are transformed through a normalizing flow producing the actual observation.

3. Background

3.1. Notation

Bold upper-case letters (*e.g.*, $\mathbf{A}, \mathbf{S}, \mathbf{P}$) denote matrices, whereas lower-case letter (*e.g.*, $\boldsymbol{\mu}, \boldsymbol{\pi}$) denote row vectors. By $\mathbf{1}$ we denote a vector consisting of ones, by $\mathbf{0}$ we denote a matrix with all entries being 0s and by $\mathbb{I}$ we denote the indicator function. By $\mathbf{1}_i = (0, \dots, 0, 1, 0, \dots, 0)$ we denote the vector with 1 on i-th position and zeros otherwise. For some matrix $\mathbf{M}$ by $\mathbf{M} \geq \mathbf{0}$ we mean that each entry is non-negative.

3.2. Hidden Markov models

Let $\{X_k\}_{k\geq 0}$ be an ergodic, time homogeneous Markov chain over hidden states $\mathcal{S} = \{s_1, \dots, s_L\}$ with a stationary distribution $\boldsymbol{\pi} = (\pi_1, \dots, \pi_L)$, and a transition matrix $\mathbf{A}$, *i.e.*, for any k we have $\mathbb{P}(X_{k+1} = s_j | X_k = s_i) = \mathbf{A}(i, j)$. Let $\{Y_k\}_{k\geq 0}$ be a sequence of random variables taking values in $\mathcal{V}$ (the *observation space*), which can be continues or discrete. Let us fix some time horizon T. Random variables $Y_{0:T} = (Y_0, \dots, Y_T)$ are independent conditionally on the state sequence $X_{0:T} = (X_0, \dots, X_T)$, *i.e.*:

$$p(Y_{0:T} = y_{0:T} | X_{0:T} = x_{0:T}) = \prod_{k=0}^{T} p(y_k | x_k), \quad (1)$$

where $p(y|x) \equiv \mathbb{P}(Y = y | X = x)$ represents the so-called *emission probabilities* for discrete observations. We will shortly write $p(y_{0:T})$ for $p(Y_{0:T} = y_{0:T})$ and similarly *e.g.*, $p(y_{0:T}|x_{0:T})$ for $p(Y_{0:T} = y_{0:T} | X_{0:T} = x_{0:T})$ to simplify the notation. Considering the continuous case, $p(y|x)$ represents the conditional emission density function for a given state x. For Gaussian HMM we assume that $p(y|x) = \mathcal{N}(y; \mu_x, \sigma_x^2)$, where the values of parameters μ_x and σ_x^2 are determined by the conditioning value x.

HMM model can be parametrized by $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\beta}\}$, where $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \dots \boldsymbol{\beta}_L\}$, and $\boldsymbol{\beta}_l$ stays behind the parameters of $p(y|l)$, for a given state l. For discrete case, $\boldsymbol{\beta}_l$ represents the parameters for categorical distribution, while for a Gaussian HMM we have $\boldsymbol{\beta}_l = \{\mu_l, \sigma_l^2\}$. The probability of observing the sequence of observations $y_{0:T}$ can be expressed as:

$$p(y_{0:T}; \boldsymbol{\theta}) = \sum_{x_{0:T} \in \mathcal{S}^{T+1}} p(y_{0:T} | x_{0:T}; \boldsymbol{\beta}) p(_{0:T}; \mathbf{A}, \boldsymbol{\pi}), \quad (2)$$

where $p(y_{0:T}|x_{0:T}; \boldsymbol{\beta})$ is given by Eq. (1), and $p(x_{0:T}; \mathbf{A}, \boldsymbol{\pi})$ can be expressed as:

$$\begin{aligned} p(x_{0:T}; \mathbf{A}, \boldsymbol{\pi}) &= \mathbb{P}(x_0) \prod_{k=1}^{T} \mathbb{P}(x_k | x_{k-1}) \\ &= \pi_{x_0} \prod_{k=1}^{T} \mathbf{A}(x_{k-1}, x_k). \end{aligned} \quad (3)$$

3.3. Normalizing Flows

Normalizing flows (Rezende & Mohamed, 2015) are generative models that can be efficiently trained via direct likelihood estimation thanks to the application of the change-of-variable formula. Practically, they utilize sequence of parametric and invertible transformations: $y = h_n \circ \dots \circ h_1(z)$. The goal of the transformation is to map z from the known, normal distribution $\mathcal{N}(z; 0, 1)$ to the more complex distribution described by a density function $f(y)$ from the observation domain. The log-probability for y can be expressed as:

$$\log f(y) = \log \mathcal{N}(z; 0, 1) - \sum_{n=1}^{N} \log \left| \det \frac{\partial h_n}{\partial z_{n-1}} \right|. \quad (4)$$

One of the main challenges while designing the normalizing flows is selection of a proper form of transformation functions h_n. The sequence of discrete transformations can be

replaced by continuous equivalent by application of Continuous Normalizing Flows (CNFs) (Grathwohl et al., 2018), where the aim is to solve differential equation of the form $\frac{dz}{dt} = g_{\boldsymbol{\beta}}(z(t), t)$, where $g_{\boldsymbol{\beta}}(z(t), t)$ represents the function of dynamics, described by parameters $\boldsymbol{\beta}$. Our goal is to find a solution of the equation in t_1, $y := z(t_1)$, assuming the given initial state $z := z(t_0)$ with a known prior. The transformation function $h_{\boldsymbol{\beta}}$ is defined as:

$$y = h_{\boldsymbol{\beta}}(z) = z + \int_{t_0}^{t_1} g_{\boldsymbol{\beta}}(z(t), t)dt. \tag{5}$$

The inverted form of the transformation can be easily computed using the formula:

$$h_{\boldsymbol{\beta}}^{-1}(y) = y - \int_{t_0}^{t_1} g_{\boldsymbol{\beta}}(z(t), t)dt. \tag{6}$$

The log-probability of y can be computed by:

$$\log f_{\boldsymbol{\beta}}(y) = \log \mathcal{N}(h_{\boldsymbol{\beta}}^{-1}(y; 0, 1)) - \int_{t_0}^{t_1} \left(\frac{dg_{\boldsymbol{\beta}}(z(t), t)}{dz(t)} \right) dt, \tag{7}$$

where $h_{\boldsymbol{\beta}}^{-1}(y) = z$.

4. FlowHMM

In this section we introduce FlowHMM - HMM variant of continuous flow capable to model the observations using complex, non-Gaussian distributions. The idea behind this approach is to model each of conditional densities $p(y|x = s_l) = f_{\boldsymbol{\beta}_l}(y)$, for each of the considered states, $s_l \in S$, using the separate CNF module. In practice, $p(y|x = s_l)$ can be calculated using formula given by Eq. (7) with the parameters $\boldsymbol{\beta}_l$ dedicated for the state s_l. The idea of our approach is illustrated in Fig. 1.

Given a sequence of observations $y_{0:T} = (y_0, \ldots, y_T)$, the model is trained by optimizing the log-likelihood of the following from $\log p(y_{0:T}; \boldsymbol{\theta})$ (see Eq. (2)), where we aim at finding the optimal values of $\mathbf{A}^*$ and $\boldsymbol{\pi}^*$, and parameters of the flows $\boldsymbol{\beta}_l^*$, such that $\boldsymbol{\theta}^* = \arg\max_\theta \log p(y_{0:T}; \boldsymbol{\theta})$. Moreover, $\mathbf{A}$ and $\boldsymbol{\pi}$ should satisfy the probability conditions.

The standard approach to solve such a formulated problem is to use the expectation-maximization (EM) algorithm. This approach is costly and may be difficult to apply to the models with complex distributions represented by CNFs. Therefore, we propose to use the co-occurrence matrix and train the model in end-to-end setting, jointly optimizing together with the parameters of the flows, $\boldsymbol{\beta}$.

4.1. Co-occurrence matrix

We assume that $\{X_k\}$ is stationary, *i.e.*, the stationary distribution $\boldsymbol{\pi}$ is also its initial distribution. In such a case, instead of using $\mathbf{A}$, the model can be equivalently represented by the state joint probabilities:

$$\begin{aligned} \mathbf{S}(i, j) &= \mathbb{P}(X_k = s_i, X_{k+1} = s_j) \\ &= \mathbb{P}(X_{k+1} = s_j | X_k = s_i)\mathbb{P}(X_k = s_i) \\ &= \mathbf{A}(i, j)\pi_i. \end{aligned} \tag{8}$$

Note that π_i can be computed as $\sum_j \mathbf{S}(i, j)$, what can be written as $\pi_i = \mathbf{1}_i \mathbf{S} \mathbf{1}^T$. For such a formulation, the HMM model can be parametrized by $\boldsymbol{\theta} = \{\mathbf{S}, \boldsymbol{\beta}_1, \ldots \boldsymbol{\beta}_L\}$. We introduce the *co-occurrence matrix* that represents the joint distribution of two consecutive observations: $\mathbf{Q}(y_1, y_2) = p(Y_{k+1} = y_2, Y_k = y_1)$, what can be rewritten as:

$$\mathbf{Q}(y_1, y_2) = \sum_{s_i, s_j \in \mathcal{S}} p(y_1|s_i)\mathbf{S}(i, j)p(y_2|s_j). \tag{9}$$

4.2. Training the model with co-occurrence matrix

Assuming the discrete set of observations, $\mathcal{V} = \{v_1, \ldots, v_M\}$, the *co-occurrence* matrix $\mathbf{Q}$ can be expressed as:

$$\mathbf{Q} = \mathbf{P}^T \mathbf{S} \mathbf{P}, \tag{10}$$

where $\mathbf{P}$ collects probabilities of all possible observations at each hidden state in a matrix $\mathbf{P}(s_i, v_j) = p(v_j|s_i)$. In this case there are M^2 possible observation pairs $(y_1, y_2) \in \mathcal{V} \times \mathcal{V}$. Note that matrices $\mathbf{Q}, \mathbf{S}, \mathbf{P}$ are of sizes $M \times M$, $L \times M$ and $L \times L$, respectively. Moreover, we have $\sum_{v_i, y_j \in \mathcal{V}} \mathbf{Q}(v_i, v_j) = \sum_{s_i, s_j} \mathbf{S}(s_i, s_j) = 1$ and $\sum_{v \in \mathcal{V}} \mathbf{P}(s_i, v) = 1$ for all $s_i \in \mathcal{S}$. In a matrix form, these can be written as $\mathbf{1}\mathbf{Q}\mathbf{1}^T = \mathbf{1}\mathbf{S}\mathbf{1}^T = 1$ and $\mathbf{P}\mathbf{1}^T = \mathbf{1}^T$ (note that $\mathbf{1}$ must be of an appropriate size). Given observations $y_{0:T}$ the matrix $\mathbf{Q}$ can be empirically estimated by:

$$\hat{\mathbf{Q}}(v_i, v_j) = \frac{1}{T} \sum_{k=0}^{T-1} \mathbb{I}(y_k = v_i)\mathbb{I}(y_{k+1} = v_j), \tag{11}$$

for all pairs $(v_i, v_j) \in \mathcal{V} \times \mathcal{V}$. The problem of training the HMM is to find such parameters (matrices $\mathbf{S}$ and $\mathbf{P}$) so that the co-occurrence matrix $\mathbf{Q}$ is close (in some sense) to the empirical co-occurrence matrix $\hat{\mathbf{Q}}$. We can formulate the problem *e.g.*, as

$$\begin{aligned} \min_{\substack{\mathbf{P} \in \mathbb{R}^{N \times M} \\ \mathbf{S} \in \mathbb{R}^{L \times L}}} \quad & ||\hat{\mathbf{Q}} - \mathbf{P}^T \mathbf{S} \mathbf{P}||^2 \\ \text{subject to} \quad & \mathbf{1}\mathbf{S}\mathbf{1}^T = 1, \mathbf{P}\mathbf{1}^T = \mathbf{1}^T, \mathbf{P} \geq \mathbf{0}, \mathbf{S} \geq \mathbf{0}. \end{aligned} \tag{12}$$

This problem formulation has a couple of advantages compared to standard likelihood-based optimization. First, the

empirical co-occurrence matrix $\hat{\mathbf{Q}}$ is independent of the sequence length T. Second, the given objective can be easily optimized using gradient-based techniques. On the other hand, the constraints for matrices $\mathbf{P}$ and $\mathbf{S}$ should be satisfied. The set $\mathcal{V}$ is discrete, while we aim to design the model for continuous observations. We are going to elaborate on these points in the following subsections.

4.3. Continuous observations and discretization.

In order to apply the given problem formulation the continuous sequence of observations should be discretized for training purposes. Let $\Gamma = (\gamma_1, \dots, \gamma_m), \gamma_i \in \mathbb{R}$, be some *grid* of size m. The grid may be predefined, created uniformly on the interval from $\gamma_1 = \min(y_{0:T})$ to $\gamma_m = \max(y_{0:T})$, or may represent the centroids of clustering approach. Having continuous observations $y_{0:T} = (y_0, \dots, y_T)$ we round each y_i to the nearest point from Γ obtaining a discretized sequence $y^{\Gamma}_{0:T} = (y^{\Gamma}_0, \dots, y^{\Gamma}_T)$, where:

$$y^{\Gamma}_i = \arg\min_{\gamma \in \Gamma} ||\gamma - y_i||^2, \quad i = 1, 2, \dots, T. \tag{13}$$

The values of the grid represents further the set $\mathcal{V}$.

4.4. Parametrization of matrices P and S.

In (Lakshminarayanan & Raich, 2010) authors proposed alternating least squares approach for optimizing (12). We will parametrize the matrix $\mathbf{S}$ by a real-valued matrix $\tilde{\mathbf{S}}$ also of size $L \times L$, using a softmax function:

$$\mathbf{S}(s_1, s_2) = \frac{\exp(\tilde{\mathbf{S}}(s_1, s_2))}{\sum_{s_i, s_j} \exp(\tilde{\mathbf{S}}(s_i, s_j))}. \tag{14}$$

Thanks to that representation, the constraints in Eq. (12) are satisfied and $\tilde{\mathbf{S}}$ can be optimized using gradient based approach.

Gaussian emission probabilities. Concerning $\mathbf{P}$, first we will consider a Gaussian version (then we denote $\mathbf{P} \equiv \mathbf{P}_{\mathcal{N}}$), the i-th row of the matrix is a density $\mathcal{N}(\mu_i, \sigma_i^2)$ evaluated at $v_1, \dots, v_M$ and normalized (by $\mathcal{N}(v; \mu, \sigma^2)$ we denote the density of $\mathcal{N}(\mu, \sigma^2)$ evaluated at v):

$$\mathbf{P}_{\mathcal{N}}(s_i, v_j) = \frac{\mathcal{N}(v_j; \mu_i, \sigma_i^2)}{\sum_{k=1}^{M} \mathcal{N}(v_k; \mu_i, \sigma_i^2)}. \tag{15}$$

Denote the parameters $\boldsymbol{\mu} = (\mu_1, \dots, \mu_L)$ and $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_L)$. Similarly, $\boldsymbol{\sigma}$ is parametrized by a real-valued vector $\tilde{\boldsymbol{\sigma}} = (\tilde{\sigma}_1, \dots, \tilde{\sigma}_L)$ in a similar fashion $\sigma_i \propto \exp(\tilde{\sigma}_i)$. Finally, the optimization problem (12) can be rewritten

$$\min_{\substack{\mathbf{S} \in \mathbb{R}^{L \times L} \\ \tilde{\boldsymbol{\sigma}}, \boldsymbol{\mu} \in \mathbb{R}^L}} ||\hat{\mathbf{Q}} - \mathbf{P}_{\mathcal{N}}^T \mathbf{S} \mathbf{P}_{\mathcal{N}}||^2. \tag{16}$$

The problem is formulated in terms of real parameters, thus gradient-based methods can be directly applied.

Flow-based emission probabilities. With each hidden state l we associate the flow model described by density function $f_{\boldsymbol{\beta}_l}$, that can be calculated from Eq. (7), where $\boldsymbol{\beta}_l$ is a set of trainable parameters of the flow. We construct $\mathbf{P}$ based on these models (in such a case $\mathbf{P} \equiv \mathbf{P}_{\boldsymbol{\beta}}$). The i-th row of the matrix is a density $f_{\boldsymbol{\beta}_i}(\cdot)$ evaluated at $v_1, \dots, v_M$ and normalized:

$$\mathbf{P}_{\boldsymbol{\beta}}(s_i, v_j) = \frac{f_{\boldsymbol{\beta}_i}(v_j)}{\sum_{k=1}^{M} f_{\boldsymbol{\beta}_i}(v_k)}. \tag{17}$$

Similarly, the optimization problem (12) becomes:

$$\min_{\boldsymbol{\beta}, \mathbf{S} \in \mathbb{R}^{L \times L}} ||\hat{\mathbf{Q}} - \mathbf{P}_{\boldsymbol{\beta}}^T \mathbf{S} \mathbf{P}_{\boldsymbol{\beta}}||^2. \tag{18}$$

However, instead of minimizing the given criterion we propose to optimize *Kullback–Leibler divergence* between distributions $\mathbf{Q}_{\beta}$, and $\hat{\mathbf{Q}}$:

$$\min_{\boldsymbol{\beta}, \mathbf{S} \in \mathbb{R}^{L \times L}} \mathcal{L}, \quad \mathcal{L} = \sum_{i,j} \mathbf{Q}_{\boldsymbol{\beta}}(i, j) \log \frac{\mathbf{Q}_{\boldsymbol{\beta}}(i, j)}{\hat{\mathbf{Q}}(i, j)}, \tag{19}$$

where $\mathbf{Q}_{\boldsymbol{\beta}} = \mathbf{P}_{\boldsymbol{\beta}}^T \mathbf{S} \mathbf{P}_{\boldsymbol{\beta}}$. We observe, that using this criterion increases the stability and convergence of the training process.

4.5. Training and inference

In this section, we introduce the procedure of training the FlowHMM. Let $y^{\text{train}}_{0:T}$ be the training set and let $y^{\text{test}}_{0:T'}$ be the test set. For the fixed m we construct the grid $\Gamma = (\gamma_1, \dots, \gamma_m)$ using one of the approaches from subsection 4.3. Next, we create the discretized training data $y^{\text{train},\Gamma}_{0:T}$ using Eq. (13). Further, we calculate the empirical co-occurrence matrix $\hat{\mathbf{Q}}$:

$$\hat{\mathbf{Q}}(\gamma_i, \gamma_j) = \frac{1}{T} \sum_{k=0}^{T-1} \mathbb{I}(y^{\text{train},\Gamma}_k = \gamma_i) \mathbb{I}(y^{\text{train},\Gamma}_{k+1} = \gamma_j), \tag{20}$$

and matrices $\mathbf{S}$ and $\mathbf{P}_{\boldsymbol{\beta}}$ (using Eqs. (14), (17)) that are further used to calculate the loss given by Eq. (7). Practically, we add Gaussian perturbations to the grid values, while calculating the matrix $\mathbf{P}$:

$$\mathbf{P}_{\boldsymbol{\beta}}(s_i, v_j) = \frac{f_{\boldsymbol{\beta}_i}(v_j + \epsilon)}{\sum_{k=1}^{M} f_{\boldsymbol{\beta}_i}(v_k + \epsilon)}, \tag{21}$$

where ϵ is a random sample from $\mathcal{N}(0, \sigma^2_{noise})$, and σ^2_{noise} is a hyperparameter of the method. These perturbations prevent from overfitting of the flow-based components caused by observing the same grid while training. Next, all of the parameters of FlowHMM are updated with gradient based approach. The procedure is repeated until convergence.

As we postulated before, our model is designed for continuous problems. Thus, during the inference stage we simply

Algorithm 1 FlowHMM - training and inference functions

Require: $y^{\text{train}}_{0:T}$, $y^{\text{test}}_{0:T'}$ — the training and testing set
Parameters: $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_L\}$ — flow parameters, $\tilde{\mathbf{S}}$ parameters representing unnormalised co-occurrence matrix.
Hyperparameters: L - number of hidden states, m - grid size, α-step size, noise variance σ^2_{noise}.

function TRAIN($y^{\text{train}}_{0:T}$, L, m, α)
  Initialize $\boldsymbol{\beta}$, and $\tilde{\mathbf{S}}$.
  Create grid $\Gamma = (\gamma_1, \dots, \gamma_m)$.
  Create discretized dataset $y^{\text{train},\Gamma}_{0:T}$ using Eq. (13).
  Calculate $\mathbf{S}$ from Eq. (14).
  while not convergent **do**
    Calculate $\mathbf{P}_{\boldsymbol{\beta}}$ from Eq. (21).
    Calculate loss function $\mathcal{L}$ from Eq. (7).
    $\tilde{\mathbf{S}} \leftarrow \tilde{\mathbf{S}} - \alpha \nabla_{\tilde{\mathbf{S}}} \mathcal{L}$
    for each $l \in \{1, \dots, L\}$ **do**
      $\boldsymbol{\beta}_l \leftarrow \boldsymbol{\beta}_l - \alpha \nabla_{\boldsymbol{\beta}_l} \mathcal{L}$
    end for
  end while
  return $\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_L, \tilde{\mathbf{S}}$
end function
function INFERENCE($y^{\text{test}}_{0:T'}$, $\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_L, \tilde{\mathbf{S}}$)
  Calculate $\mathbf{S}$ from Eq. (14).
  Calculate $\mathbf{A}$ and $\boldsymbol{\pi}$ from $\mathbf{S}$.
  Calculate $p(y^{\text{test}}_{0:T}; \boldsymbol{\theta} = \{\mathbf{A}, \boldsymbol{\pi}, \boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_L)\}$ from Eq. (1) using forward algorithm.
  return $p(y^{\text{test}}_{0:T}; \boldsymbol{\theta})$
end function

calculate $\mathbf{A}$ and $\boldsymbol{\pi}$ from $\mathbf{S}$ and apply forward procedure on the test data $y^{\text{test}}_{0:T'}$ to calculate $p(y^{\text{test}}_{0:T'}; \boldsymbol{\theta})$. The procedures of training and inference are presented in Algorithm 1.

5. Experimental results

Evaluation. Our main flow-based model is denoted by H^{Flow}, the trained parameters are denoted as $\boldsymbol{\beta}^{\text{opt}}_1, \dots, \boldsymbol{\beta}^{\text{opt}}_L, \tilde{\mathbf{S}}^{\text{opt}}$. We will compare our results with H^{Gauss} model – a model with Gaussian observation densities, we use `hmmlearn` library[1]. In all the examples, the H^{Flow} is trained on $y^{\text{train},\Gamma}_{0:T}$, whereas H^{Gauss} is trained on $y^{\text{train}}_{0:T}$. For a model H we compute normLL – the normalized log-likelihood of continuous observations $y_{0:T'}$, *i.e.* $\log \mathcal{L}(y_{0:T'})/(T'+1)$. To compute it for H^{Flow} we used the standard forward algorithm.

Moreover, for synthetic examples (for which we know the observations densities $p(y|s_i)$) we will also determine how well the model learned the observation densities. Note that we may evaluate $f_{\boldsymbol{\beta}^{\text{opt}}_i}(y)$ at finite number of points $y \in \mathbb{R}$, we must thus do it on some grid. We will use another, more dense, uniform grid $\Gamma' = (\gamma'_1, \dots, \gamma'_r)$, where $\gamma'_1 = \gamma_1, \gamma'_r = \gamma_m$. We take $r = 10m$ in all the examples. For two continuous distributions g_1 and g_2 we compute the total variation distance as

$$d_{tv}(g_1, g_2) = \frac{1}{2} \sum_{k=1}^{r} |g_1(\gamma'_k) - g'_2(\gamma_k)|$$

(where we actually normalize g_i's to be distributions on Γ'). To compare distributions $p(\cdot|s_i)$ with with $f_{\boldsymbol{\beta}^{i,\text{opt}}}(\cdot)$, $i = 1, \dots, L$ (*i.e.*, with distribution learnt by H^{Flow}), we compute the mean of total variation distances

$$d^{\text{Flow}}_{tv} = \frac{1}{L} \sum_{i=1}^{L} d_{tv}(p(\cdot|s_i), f_{\boldsymbol{\beta}^{\text{opt}}_i})$$

$$= \frac{1}{2L} \sum_{i=1}^{L} \sum_{k=1}^{r} |p(\gamma'_k|s_i) - f_{\boldsymbol{\beta}^{\text{opt}}_i}(\gamma'_k)|.$$

Similarly, for H^{Gauss} we compute $d^{\text{Gauss}}_{tv} = \frac{1}{L} \sum_{i=1}^{L} d_{tv}(p(\cdot|s_i), \mathcal{N}(\cdot; \mu^{\text{opt}}_i, (\sigma^{\text{opt}}_i)^2))$, where $\mu^{\text{opt}}_i, (\sigma^{\text{opt}}_i)^2$ are the parameters learnt by H^{Gauss}.

Experimental settings We use ADAM (Kingma & Ba, 2014) as an optimizer with $lr = 0.01$, default betas $(0.9, 0.999)$, and weight decay equal 0.0001. In all experiments, we use two blocks of CNFs with *concatsquash* layers, batch norm, and the configuration of the hidden neurons equal $16 - 16$.[2]

Synthetic sequences. In the examples we will use one of the following two transition matrices

$$\mathbf{A}_1 = \begin{pmatrix} 0.0 & 0.9 & 0.1 \\ 0.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 \end{pmatrix}, \quad \mathbf{A}_2 \begin{pmatrix} 0.4 & 0.2 & 0.4 \\ 0.25 & 0.5 & 0.25 \\ 0.4 & 0.2 & 0.4 \end{pmatrix}.$$

The matrix $\mathbf{A}_1$ was used in (Lakshminarayanan & Raich, 2010) (see Eq. (11) therein). As observation densities we will use: uniform distribution $\mathcal{U}(a, b)$, normal distribution $\mathcal{N}(\mu, \sigma^2)$ or beta distribution Beta(α, β) with shape parameters $\alpha > 0, \beta > 0$. The latter has the density function $p(y) \propto y^{\alpha-1}(1-y)^{\beta-1}$ for $y \in (0, 1)$.

In the synthetic examples below we take $T = T'$, *i.e.*, the same lengths of training and testing set. We will comment on the results of Example 1, 2a and 2b after introducing them.

Example 1. We reconsider example from (Lakshminarayanan & Raich, 2010), which is a HMM with $L = 3$ hidden states, the transition matrix $\mathbf{A}_1$ and with the following observation densities: $p(\cdot|s_1) \sim \mathcal{N}(11, 2)$, $p(\cdot|s_1) \sim \mathcal{N}(16, 3)$ and $p(\cdot|s_3) \sim \mathcal{U}(16, 26)$.

[1] https://github.com/hmmlearn

[2] We make the code available at http://url.blinded.during.review.period

	T	Example 1		Example 2a		Example 2b	
		Gauss	Flow	Gauss	Flow	Gauss	Flow
normLL	10^3	**-2.198**	-2.827	**0.164**	0.107	0.093	**0.157**
	10^4	-2.188	**-2.179**	0.174	**0.335**	0.089	**0.299**
	10^5	-2.187	**-2.161**	0.177	**0.348**	0.089	**0.324**
d_{tv}	10^3	**0.086**	0.337	**0.285**	0.394	–	–
	10^4	0.071	**0.046**	0.280	**0.123**	–	–
	10^5	0.067	**0.035**	0.283	**0.089**	–	–

Table 1: Normalized log-likelihoods and total variation distances of H^{Flow} and H^{Gauss} for synthetic Examples 1, 2a-b.

We take $m = 30$ and the grid being integers $\gamma = \{0, 1 \dots, m\}$ (to be consistent with discretization in (Lakshminarayanan & Raich, 2010)). We train the models H^{Flow} and H^{Gauss} with 3 hidden states.

Example 2a. We take a model with $L = 3$ hidden states, the transition matrix $\mathbf{A}_2$ and following observation densities: $p(\cdot|s_1) \sim \text{Beta}(7, 1.1)$, $p(\cdot|s_2) \sim \mathcal{U}(0.2, 0.4)$ and $p(\cdot|s_3) \sim \mathcal{N}(0.6, 0.0046)$. We take the uniform grid $\Gamma = (\gamma_1, \dots, \gamma_m)$ of size $m = 30$ with $\gamma_1 = \min(y^{\text{train}}_{0:T})$ and $\gamma_k = \max(y^{\text{train}}_{0:T})$. We train the models H^{Flow} and H^{Gauss} with 3 hidden states.

Example 2b. This is actually Example 2a, however we train models H^{Flow} and H^{Gauss} with 2 hidden states.

For each example we sample train and test observations of length $T \in \{10^3, 10^4, 10^5\}$. We train H^{Flow} model using 500 (Example 1) and 1000 (Example 2) epochs. For each case (*i.e.*, fixed example and T) we perform 10 simulations and report means of normLLs and total variation distances in Table 1. For observations of length $T \geq 10^3$ model H^{Flow} clearly outperforms H^{Gauss} in all the examples. Note that in these examples we had a grid of size $m = 30$, thus we estimated the matrix with 900 entries (number of pairs). It is intuitively clear that $T = 10^3$ observations is not enough then. In Fig. 3 the distributions learned by our model H^{Flow} are depicted (from single simulations). As we can see, for $T \geq 10^4$ the small values of total variation distances from Table 1 are confirmed. We encourage the reader to confront Fig. 3 (d) bottom with Fig. 2 and Fig. 4 from (Lakshminarayanan & Raich, 2010). In Fig. 2 the distributions learned by H^{Flow} and H^{Gauss} models with $L = 2$ hidden states (recall, they were sampled from HMM with 3 hidden states) are depicted. Since flow models are capable of fitting multimodal distributions, H^{Flow} easily outperforms H^{Gauss}.

Example 3. We shortly report the results (concerning mainly execution time) of optimization of (16), *i.e.*, the model with Gaussian observations, call it $H^{\mathcal{N}}$. As a HMM we have a model with transition matrix $\mathbf{A}_1$ and Gaussian observation densities: $p(\cdot|s_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$ with $(\mu_1, \sigma_1^2) = (4, 2), (\mu_2, \sigma_2^2) = (9, 3), (\mu_3, \sigma_3^2) = (17, 3)$.

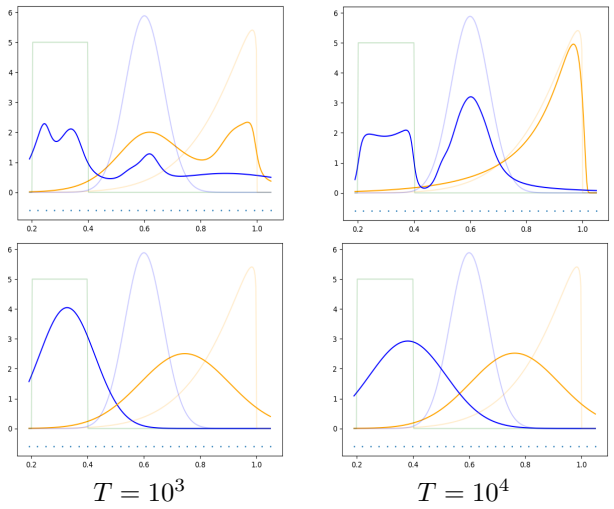

Figure 2: Results for Example 2b (three original distributions shaded) for model H^{Flow} (top) and H^{Gauss} (bottom)

L	S&P 500		Dow Jones		Air pressure	
	Gauss	Flow	Gauss	Flow	Gauss	Flow
2	-9.789	-5.070	-11.594	-6.620	-0.704	0.823
3	-10.073	-4.587	-12.348	-6.865	-0.742	0.861
4	-11.630	-4.659	-13.594	-6.841	-0.758	0.780

Table 2: Normalized log-likelihoods for real datasets (Examples 4 and 5).

We sampled 500k observations and trained $H^{\mathcal{N}}$ on a uniform grid of size $m = 30$. We also trained the H^{Gauss} model. Densities learned by both models are very similar, the parameters learned by H^{Gauss} are $(\mu_1, \sigma_1^2) = (3.997, 1.997), (\mu_2, \sigma_2^2) = (8.992, 2.995), (\mu_3, \sigma_3^2) = (16.994, 3.008)$, whereas the parameters learned by $H^{\mathcal{N}}$ are: $(\mu_1, \sigma_1^2) = (4.002, 2.093), (\mu_2, \sigma_2^2) = (8.992, 3.084), (\mu_3, \sigma_3^2) = (16.989, 3.094)$. However it took 12.94s for H^{Gauss}, whereas it took 3.55s for $H^{\mathcal{N}}$.

Real datasets.

Example 4. S&P 500 and Dow Jones are popular measures of market performance. We retrieved data (`data.world`, accessed 2022) for a period of 9/1977-8/2017 that consist of 2082 points. To remove trend we applied a standard difference transform, *i.e.*, we consider time series $y'_k = y_k - y_{k-1}$. We trained models H^{Flow} and H^{Gauss} on $T = 1000$ observations (and computed normLLs for the remaining data) with $L = \{2, 3, 4\}$ hidden states. For H^{Flow} we used a uniform grid of size $m = 30$, it was trained for 500 epochs. The results are shown in Table 2, learned observation distributions for $L \in \{3, 4\}$ are depicted in Fig. 4. Clearly H^{Flow} outperformed H^{Gauss} in any case, even having not large number of observations – note that for $m = 30$ we have 900 pairs (and just $T = 1000$ observations).

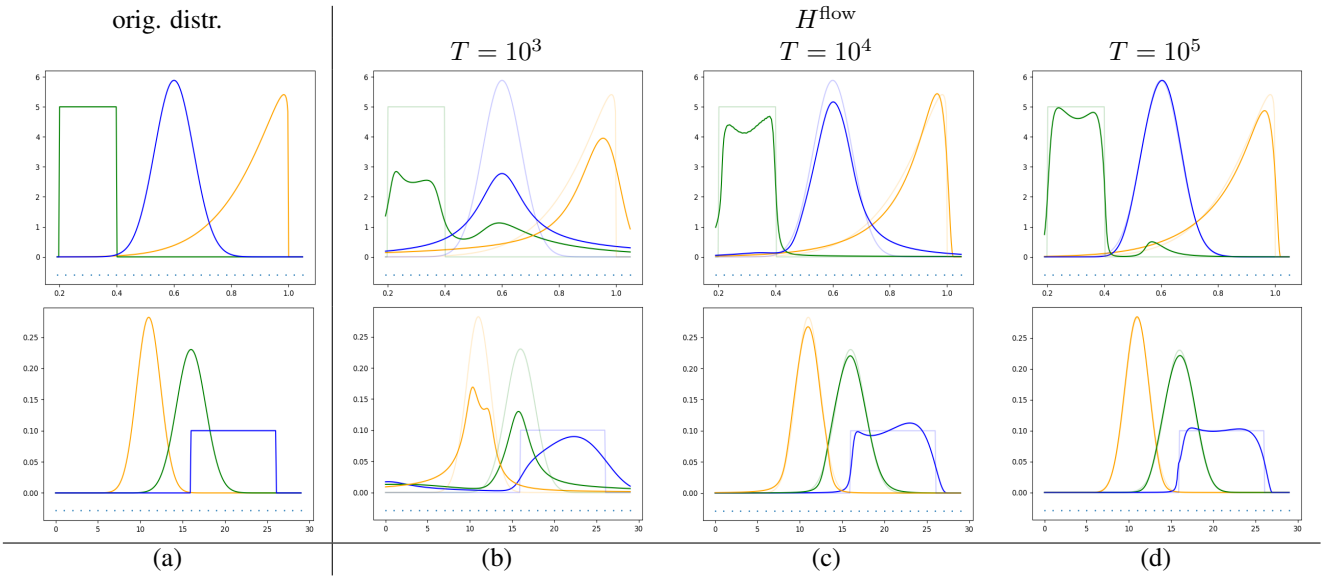

Figure 3: Original distributions $p(\cdot|s_i), i = 1, 2, 3$ (a), distributions learned by H^{Flow} on $T = 10^3$ (b), $T = 10^4$ (c) and $T = 10^5$ (d) observations for Example 1 (top row) and Example 2a (bottom row).

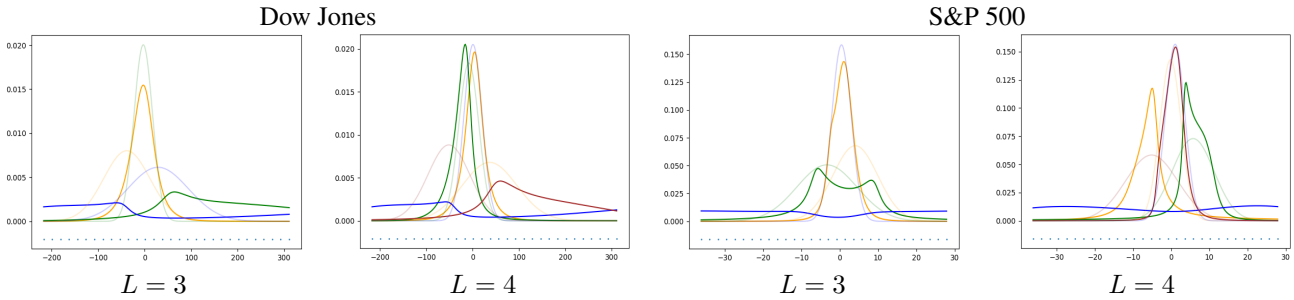

Figure 4: Distributions learned by H^{Flow} (colored) and H^{Gauss} (grayed-out) for Dow Jones and S&P 500 datasets (models with $L \in \{3, 4\}$ hidden states).

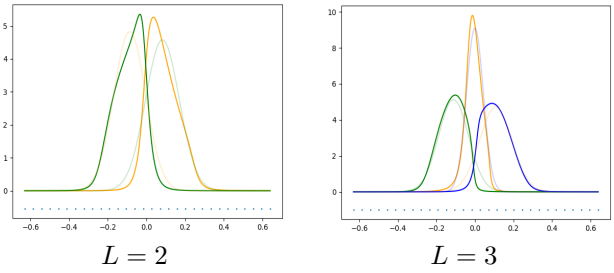

Figure 5: Distributions learned by H^{Flow} (colored) and H^{Gauss} (grayed-out) for Example 5 (models with $L \in \{2, 3\}$ hidden states).

Example 5. We used mean air pressure from (AmeriGEOSS, 2019) dataset collected between 02/15/18 and 02/28/19. We applied the standard difference transform obtaining 54 531 observations, from which we trained H^{Flow} and H^{Gauss} with $L \in \{2, 3, 4\}$ hidden states on the first $T = 40000$ samples and tested them on the remaining observations. Similarly, we used grid of size $m = 30$ and performed the training for 500 epochs. Table 2 shows the results, where H^{Flow} outperforms H^{Gauss} by a large margin. Fig. 5 shows that the learned observation densities for $L \in \{2, 3\}$ are non-Gaussian (for $L = 3$ the density drawn in blue is *more flat* then the Gaussian one).

6. Conclusions and limitations

In this work, we proposed a continuous hidden Markov model that leverages deep flow-based network architectures to model complex, non-Gaussian distributions. Although our approach outperforms several baselines and competing approaches, it relies on the co-occurrence matrix that can be only computed with the assumption that the Markov chain on a set of hidden states is stationary - a limitation not present in the EM-based approaches. Addressing this constraint, *e.g.* by re-computing temporally stationary co-occurrence matrix during training, is part of our future work. So is validating our model on multivariate time series or distributions, where our flexible approach can potentially offer more benefits over the existing works.

References

Ailliot, P., Thompson, C., and Thomson, P. Space–time modelling of precipitation by using a hidden markov model and censored gaussian distributions. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 58(3): 405–426, 2009.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

Bourlard, H. and Wellekens, C. J. Links between markov models and multilayer perceptrons. *IEEE Transactions on pattern analysis and machine intelligence*, 12(12): 1167–1178, 1990.

Cavanaugh, J. E. and Neath, A. A. The Akaike information criterion: Background, derivation, properties, application, interpretation, and refinements. *WIREs Computational Statistics*, 11(3):e1460, 2019. doi: https://doi.org/10.1002/wics.1460. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.1460.

Dannemann, J. Semiparametric hidden markov models. *Journal of computational and graphical statistics*, 21(3): 677–692, 2012.

Dannemann, J., Holzmann, H., and Leister, A. Semiparametric hidden markov models: identifiability and estimation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):418–425, 2014.

Darwin, O. and Kiefer, S. Equivalence of hidden markov models with continuous observations. *arXiv preprint arXiv:2009.12978*, 2020.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., and Januschowski, T. Normalizing kalman filters for multivariate time series analysis. In *NeurIPS*, 2020.

Du, L., Chen, M., Lucas, J., and Carin, L. Sticky hidden markov modeling of comparative genomic hybridization. *IEEE Transactions on Signal Processing*, 58(10):5353–5368, 2010. doi: 10.1109/TSP.2010.2053033.

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. An hdp-hmm for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, pp. 312–319, 2008.

Gales, M. and Young, S. The application of hidden markov models in speech recognition. 2008.

Ghosh, A., Honoré, A., Liu, D., Henter, G. E., and Chatterjee, S. Robust classification using hidden markov models and mixtures of normalizing flows. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2020. doi: 10.1109/MLSP49062.2020.9231775.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv*, 2018.

Hsu, D., Kakade, S. M., and Zhang, T. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

Huang, K., Fu, X., and Sidiropoulos, N. Learning hidden markov models from pairwise co-occurrences with application to topic modeling. In *International Conference on Machine Learning*, pp. 2068–2077. PMLR, 2018.

Ilhan, F., Karaahmetoglu, O., Balaban, I., and Kozat, S. S. Markovian rnn: An adaptive time series prediction network with hmm-based switching for nonstationary environments. *arXiv preprint arXiv:2006.10119*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kontorovich, A., Nadler, B., and Weiss, R. On learning parametric-output hmms. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. III–702–III–710. JMLR.org, 2013.

Lakshminarayanan, B. and Raich, R. Non-negative matrix factorization for parameter estimation in hidden markov models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 89–94, 2010. doi: 10.1109/MLSP.2010.5589231.

Liu, Y.-Y., Li, S., Li, F., Song, L., and Rehg, J. M. Efficient learning of continuous-time hidden markov models for disease progression. *Advances in neural information processing systems*, 28:3599, 2015.

Mamon, R. S. and Elliott, R. J. *Hidden Markov models in finance*, volume 4. Springer, 2007.

Mattila, R., Rojas, C. R., Krishnamurthy, V., and Wahlberg, B. Identification of hidden markov models using spectral learning with likelihood maximization. In *2017 IEEE 56th annual conference on decision and control (CDC)*, pp. 5859–5864. IEEE, 2017.

Mattila, R., Rojas, C., Moulines, E., Krishnamurthy, V., and Wahlberg, B. Fast and consistent learning of hidden Markov models by incorporating non-consecutive

correlations. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6785–6796. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/mattila20a.html.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *ICML*. PMLR, 2015.

Salazar, J., Robinson, M., and Azimi-Sadjadi, M. R. A hybrid hmm-neural network with gradient descent parameter training. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 2, pp. 1086–1091. IEEE, 2003.

Schwarz, G. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461 – 464, 1978. doi: 10.1214/aos/1176344136. URL https://doi.org/10.1214/aos/1176344136.

Sicking, J., Pintz, M., Akila, M., and Wirtz, T. DenseHMM: Learning hidden markov models by learning dense representations. *CoRR*, abs/2012.09783, 2020. URL https://arxiv.org/abs/2012.09783.

AmeriGEOSS. Wind-measurementds in Papua New Guinea, 2019. data retrieved from https://data.amerigeoss.org/dataset/857a9652-1a8f-4098-9f51-433a81583387/resource/a8db6cec-8faf-4c8e-aee2-4fb45d1e6f14.

data.world. DowJones and S&P500 datasets, accessed 2022. https://data.world/chasewillden/stock-market-from-a-high-level/workspace/.

Tran, K., Bisk, Y., Vaswani, A., Marcu, D., and Knight, K. Unsupervised neural hidden markov models. *arXiv preprint arXiv:1609.09007*, 2016.

Vidyasagar, M. *Hidden Markov Processes: Theory and Applications to Biology*. Princeton University Press, 2014.

Xia, L., Chen, C.-C., and Aggarwal, J. K. View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 20–27, 2012. doi: 10.1109/CVPRW.2012.6239233.

Yu, H. A novel semiparametric hidden markov model for process failure mode identification. *IEEE Transactions on Automation Science and Engineering*, 15(2):506–518, 2018. doi: 10.1109/TASE.2016.2636292.