Partially Variational Expectation-Maximization for Deep Generative Models

Author 1*
Department of YYY, University of XXX
and
Author 2
Department of ZZZ, University of WWW

November 12, 2025

Abstract

Abstrct.

Keywords: 7 or fewer keywords

1 Introduction

2 Methods

2.1 From EM to Partially Variational EM

2.1.1 EM

We begin by reviewing the classical Expectation-Maximization (EM) algorithm. Suppose $\mathbf{y}$ denotes the observed variable, $\mathbf{x}$ is the latent (or missing) variable, and θ represents the model parameters. The goal of the EM algorithm is to maximize the observed-data likelihood:

$$L(\theta \mid \mathbf{y}) = p_\theta(\mathbf{y}).$$

*The authors gratefully acknowledge

In many cases, however, we only have access to the complete-data likelihood $L(\theta \mid \mathbf{y}, \mathbf{x}) = p_\theta(\mathbf{y}, \mathbf{x})$, and computing the marginal likelihood by integrating out the latent variable is intractable:

$$L(\theta \mid \mathbf{y}) = \int p_\theta(\mathbf{y}, \mathbf{x})\, d\mathbf{x}.$$

To address this difficulty, the EM algorithm introduces a surrogate objective, known as the Q-function, which is iteratively updated and maximized. This surrogate ensures that each iteration monotonically increases the observed-data likelihood, thereby guaranteeing convergence.

At iteration t, the Q-function is defined as:

$$Q(\theta \mid \theta^{(t)}) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x} \mid \mathbf{y}, \theta^{(t)})} \left[\log L(\theta \mid \mathbf{y}, \mathbf{x}) \right],$$

where $p(\mathbf{x} \mid \mathbf{y}, \theta^{(t)})$ is the posterior distribution of the latent variable under the current parameter estimate $\theta^{(t)}$. The parameters are then updated by maximizing the Q-function:

$$\theta^{(t+1)} = \arg\max_\theta Q(\theta \mid \theta^{(t)}).$$

2.1.2 Variational EM

We next consider the variational EM algorithm. In situations where the posterior distribution $p(\mathbf{x} \mid \mathbf{y}, \theta)$ is intractable to evaluate or sample from, the standard EM algorithm cannot be applied directly. To overcome this issue, the variational EM algorithm replaces the true posterior with a tractable variational distribution $q(\mathbf{x})$ that approximates $p(\mathbf{x} \mid \mathbf{y}, \theta)$. The algorithm proceeds by maximizing a lower bound on the marginal log-likelihood, known as the evidence lower bound (ELBO), which is defined as

$$\log p_\theta(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{x})} \left[\log p_\theta(\mathbf{y}, \mathbf{x}) \right] - \mathbb{E}_{q(\mathbf{x})} \left[\log q(\mathbf{x}) \right] =: \mathcal{L}(q, \theta).$$

At each iteration, the variational EM algorithm alternates between two steps. First, the variational distribution $q(\mathbf{x})$ is updated to maximize the ELBO with respect to q, using the current estimate of the parameters $\theta^{(t)}$:

$$q^{(t)}(\mathbf{x}) = \arg\max_q \mathcal{L}(q, \theta^{(t)}).$$

This step corresponds to approximating the posterior distribution by minimizing the Kullback–Leibler divergence between $q(\mathbf{x})$ and $p(\mathbf{x} \mid \mathbf{y}, \theta^{(t)})$. Then, with the variational distribution fixed,

the model parameters are updated by maximizing the ELBO with respect to θ:

$$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{L}(q^{(t)}(\mathbf{x}), \theta).$$

By replacing the exact posterior with a flexible approximation, variational EM extends the applicability of EM to models with complex latent structures, for which exact inference is computationally infeasible.

2.1.3 Partially Variational EM

In the partially variational EM framework, we explicitly partition the latent variables into two subsets, denoted by $\mathbf{x}$ and $\mathbf{z}$, while the observed variable remains $\mathbf{y}$. The key idea is to apply different inference strategies to these two parts: we approximate the posterior of $\mathbf{z}$ using a variational distribution, while retaining the exact conditional posterior for $\mathbf{x}$.

Specifically, we assume that the approximate posterior factorizes as

$$q(\mathbf{x}, \mathbf{z} \mid \mathbf{y}, \theta^{(t)}) = q(\mathbf{z} \mid \mathbf{y})\, p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(t)}),$$

where $q(\mathbf{z} \mid \mathbf{y})$ is a variational distribution, and $p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(t)})$ is the exact conditional posterior under the current model parameters. Under this factorization, the evidence lower bound (ELBO) at iteration t becomes

$$\mathcal{L}^{(t)}(q, \theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{y})}\left[\mathbb{E}_{p(\mathbf{x}|\mathbf{z},\mathbf{y},\theta^{(t)})}\left[\log \frac{p(\mathbf{y}, \mathbf{x}, \mathbf{z} \mid \theta)}{q(\mathbf{z} \mid \mathbf{y})\, p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(t)})}\right]\right],$$

This partially variational approach defines a hybrid inference scheme in which only a subset of the latent variables is approximated variationally. By doing so, it allows us to leverage the high representational capacity of deep generative models for modeling complex data, while preserving interpretability and analytical tractability for a subset of model components. A simplified convergence analysis for the partially variational EM algorithm is provided in the supporting material.

The partially variational EM framework is general and can be instantiated in a variety of probabilistic models. In the following sections, we demonstrate its applicability to two representative classes of latent variable models: mixture models and hidden Markov models. In both cases, we incorporate expressive deep generative components (VAE and DPM) to model complex data distributions, while maintaining structured latent variables that admit exact conditional inference.

This design allows us to learn rich representations from data while preserving interpretability and tractable learning dynamics for key model components.

2.2 Mixture Models with Deep Generative Components

Traditional mixture models usually consider simplex distributions as components. While it is computational efficient, it has a limit on the fitting power. Here we consider deep generative models as the components. Suppose $\mathbf{y}$ denotes the observed variable, $\mathbf{x} \in \{1, \ldots, K\}$ represents the cluster label. Let $\Pi = (\pi_1, \pi_2, \cdots, \pi_K)$ be the prior distribution for the cluster label. For deep generative components, we additionally introduce a latent variable $\mathbf{z}$.

2.2.1 General framework of mVAE

In VAE, we have an encoder $q_\phi(\mathbf{z} \mid \mathbf{y})$ and a decoder $p_\theta(\mathbf{y} \mid \mathbf{z}, \mathbf{x})$. Here $\mathbf{x}$ is treated as the condition. The parameter ϕ is related to the encoder, the parameter θ is related to the decoder. We have the ELBO at t-th iteration:

$$\mathcal{L}^{(t)}(\phi, \theta, \Pi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{y})}\left[\mathbb{E}_{p(\mathbf{x}|\mathbf{z},\mathbf{y},\theta^{(t)},\Pi^{(t)})}\left[\log \frac{p(\mathbf{y}, \mathbf{x}, \mathbf{z} \mid \theta, \Pi)}{q_\phi(\mathbf{z} \mid \mathbf{y})\, p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(t)}, \Pi^{(t)})}\right]\right],$$

We define

$$\mathcal{J}^{(t)}(\mathbf{x}, \mathbf{z}, \mathbf{y}, \phi, \theta, \Pi) := \log \frac{p(\mathbf{y}, \mathbf{x}, \mathbf{z} \mid \theta, \Pi)}{q_\phi(\mathbf{z} \mid \mathbf{y}) p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(t)}, \Pi^{(t)})}$$

In VAE, it is difficult to calculate the expectation related to the complex variational distribution $q_\phi(\mathbf{z} \mid \mathbf{y})$. And because the expectation is related to the parameter we want to optimize, it cannot directly use Monte Carlo to estimate the expectation and calculate the gradient:

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{y})}\left[\mathcal{J}^{(t)}(\mathbf{x}, \mathbf{z}, \mathbf{y}, \phi, \theta, \Pi)\right] \neq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{y})}\left[\nabla_\phi \mathcal{J}^{(t)}(\mathbf{x}, \mathbf{z}, \mathbf{y}, \phi, \theta, \Pi)\right]$$

Here we utilize reparameterization trick. We express the latent variable $\mathbf{z}$ as a deterministic variable $\mathbf{z} = g_\phi(\boldsymbol{\epsilon}, \mathbf{y})$, where $\boldsymbol{\epsilon}$ is an auxiliary variable with independent marginal $p(\boldsymbol{\epsilon})$, and $g_\phi(\cdot)$

is some vector-valued function parameterized by ϕ:

$$\begin{aligned}
&\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{y})}\left[\mathbb{E}_{p(\mathbf{x}|\mathbf{z},\mathbf{y},\theta^{(t)},\Pi^{(t)})}\left[\mathcal{J}^{(t)}(\mathbf{x},\mathbf{z},\mathbf{y},\phi,\theta,\Pi)\right]\right] \\
&= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\phi \mathbb{E}_{p(\mathbf{x}|g_\phi(\boldsymbol{\epsilon},\mathbf{y}),\mathbf{y},\theta^{(t)},\Pi^{(t)})}\left[\mathcal{J}^{(t)}(\mathbf{x},g_\phi(\boldsymbol{\epsilon},\mathbf{y}),\mathbf{y},\phi,\theta,\Pi)\right]\right] \\
&= \mathbb{E}_{p(\boldsymbol{\epsilon})}\Big[\mathbb{E}_{p(\mathbf{x}|g_\phi(\boldsymbol{\epsilon},\mathbf{y}),\mathbf{y},\theta^{(t)},\Pi^{(t)})}\left[\nabla_\phi \mathcal{J}^{(t)}(\mathbf{x},g_\phi(\boldsymbol{\epsilon},\mathbf{y}),\mathbf{y},\phi,\theta,\Pi)\right. \\
&+ \left.\mathcal{J}^{(t)}(\mathbf{x},g_\phi(\boldsymbol{\epsilon},\mathbf{y}),\mathbf{y},\phi,\theta,\Pi)\nabla_\phi \log p(\mathbf{x} \mid g_\phi(\boldsymbol{\epsilon},\mathbf{y}),\mathbf{y},\theta^{(t)},\Pi^{(t)})\right]\Big]
\end{aligned}$$

Note that the inner expectation is also related to the parameter ϕ. When we take gradients inside the expectation, it also raises another term in addition to the gradient of $\mathcal{J}$. Given this, we also utilize reparameterization trick to $\mathbf{x}$. We express the latent variable $\mathbf{x}$ as a deterministic variable $\mathbf{x} = h_t(\boldsymbol{\xi},\mathbf{z},\mathbf{y})$, where $\boldsymbol{\xi}$ is an auxiliary variable with independent marginal $p(\boldsymbol{\xi})$, and $h_t(\cdot)$ is some vector-valued function parameterized by $\theta^{(t)}$ and $\Pi^{(t)}$:

$$\nabla_\phi \mathcal{L}^{(t)}(\phi,\theta,\Pi) = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\mathbb{E}_{p(\boldsymbol{\xi})}\left[\nabla_\phi \mathcal{J}^{(t)}(h_t(\boldsymbol{\xi},g_\phi(\boldsymbol{\epsilon},\mathbf{y}),\mathbf{y}),g_\phi(\boldsymbol{\epsilon},\mathbf{y}),\mathbf{y},\phi,\theta,\Pi)\right]\right],$$

where $p(\boldsymbol{\xi})$ is usually a Gumbel distribution for the discrete latent variable. We use Monte Carlo to estimate:

$$\nabla_\phi \tilde{\mathcal{L}}^{(t)}(q,\theta,\Pi) = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{M}\sum_{m=1}^{M}\left[\nabla_\phi \mathcal{J}^{(t)}(\mathbf{x}^{(m,l)},\mathbf{z}^{(l)},\mathbf{y},\phi,\theta,\Pi)\right],$$

where

$$\begin{aligned}
\mathbf{z}^{(l)} &= g_\phi(\boldsymbol{\epsilon}^{(l)},\mathbf{y}), \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}), \quad l = 1,2,\cdots,L \\
\mathbf{x}^{(m,l)} &= h_t(\boldsymbol{\xi}^{(m,l)},g_\phi(\boldsymbol{\epsilon}^{(l)},\mathbf{y}),\mathbf{y}), \quad \boldsymbol{\xi}^{(m,l)} \sim p(\boldsymbol{\xi}), \quad m = 1,2,\cdots,M.
\end{aligned}$$

For convenience, the updates of θ and Π can also utilize the above reparameterization method to compute gradients.

2.2.2 Specific example of mVAE

In this subsection, we consider a specific setting of mVAE. Let the prior of latent variable $\mathbf{z}$ be standard Gaussian distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. Correspondingly, the variational distribution (encoder) is a multivariate Gaussian with a diagonal covariance structure:

$$q_\phi(\mathbf{z} \mid \mathbf{y}) = \mathcal{N}(\mathbf{z};\boldsymbol{\mu}_\phi(\mathbf{y}),\boldsymbol{\sigma}^2_\phi(\mathbf{y})\mathbf{I}),$$

where the mean and s.d. of the approximate posterior, $\boldsymbol{\mu}_\phi(\mathbf{y})$ and $\boldsymbol{\sigma}^2_\phi(\mathbf{y})$ are outputs of the encoding MLP. And we set:

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}, \theta) = \mathrm{Cat}(\mathbf{y}; \boldsymbol{f}_\theta(\mathbf{x}, \mathbf{z}))$$

where $\boldsymbol{f}_\theta(\mathbf{x}, \mathbf{z})$ is the output of the decoding MLP (categorical probabilities), $\mathbf{x}$ is the condition.

Therefore, we have

$$p(\mathbf{y}, \mathbf{x}, \mathbf{z} \mid \theta, \Pi) = p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}, \theta) p(\mathbf{x} \mid \Pi) p(\mathbf{z}) = \pi_{\mathbf{x}} \mathrm{Cat}(\mathbf{y}; \boldsymbol{f}_\theta(\mathbf{x}, \mathbf{z})) \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$$

And we have

$$p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(t)}, \Pi^{(t)}) \propto p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}, \theta) p(\mathbf{x} \mid \Pi).$$

Then

$$p(\mathbf{x} = k \mid \mathbf{z}, \mathbf{y}, \theta^{(t)}, \Pi^{(t)}) = \frac{s_k^{(t)}(\mathbf{z})}{\sum_{k'=1}^{K} s_{k'}^{(t)}(\mathbf{z})}, \quad \text{where} \quad s_k^{(t)}(\mathbf{z}) = \pi_k^{(t)} \mathrm{Cat}(\mathbf{y}; \boldsymbol{f}_{\theta^{(t)}}(k, \mathbf{z}))$$

Then we can calculate $\mathcal{J}^{(t)}$:

$$\mathcal{J}^{(t)}(\mathbf{x}, \mathbf{z}, \mathbf{y}, \phi, \theta, \Pi) = \log \frac{p(\mathbf{y}, \mathbf{x}, \mathbf{z} \mid \theta, \Pi)}{q_\phi(\mathbf{z} \mid \mathbf{y}) p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(t)}, \Pi^{(t)})}$$

when $\mathbf{x}$ and $\mathbf{z}$ are known. However, as stated above, we cannot directly use the Monte Carlo to estimate the expectation when considering gradients. Here for the setting we have:

$$\mathbf{z}^{(l)} = \boldsymbol{\mu}_\phi(\mathbf{y}) + \boldsymbol{\sigma}_\phi(\mathbf{y}) \odot \boldsymbol{\epsilon}^{(l)}, \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad l = 1, 2, \cdots, L.$$

For the discrete label $\mathbf{x}$, we can use Gumbel-max trick to re-parameterize:

$$\mathbf{x}^{(m,l)} = \arg\max_k \left[\log s_k^{(t)}(\mathbf{z}^{(l)}) + \xi_k^{(m,l)} \right], \quad \xi_k^{(m,l)} \sim \mathrm{Gumbel}(0, 1), \quad m = 1, 2, \cdots, M.$$

The proof that the distribution remain the same after re-reparameterization ($\mathbf{x}^{(m,l)} \sim p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(t)}, \Pi^{(t)})$) is provided in the supplementary. However, argmax cannot track the gradients for PyTorch. In practical, we use Gumbel softmax:

$$\mathbf{x}^{(m,l)} = \frac{\exp\left(\left[\log s_k^{(t)}(\mathbf{z}^{(l)}) + \xi_k^{(m,l)} \right] / \tau \right)}{\sum_j \exp\left(\left[\log s_j^{(t)}(\mathbf{z}^{(l)}) + \xi_j^{(m,l)} \right] / \tau \right)}, \quad \xi_k^{(m,l)} \sim \mathrm{Gumbel}(0, 1), \quad m = 1, 2, \cdots, M,$$

when τ tend to be zero, the $\mathbf{x}^{(m,l)}$ is one-hot encoding label. In applications, we will gradually decrease the τ. Then we can use the estimated gradient to update parameters:

$$\nabla_{\phi,\theta,\Pi} \tilde{\mathcal{L}}^{(t)}(q, \theta, \Pi) = \frac{1}{L} \sum_{l=1}^{L} \frac{1}{M} \sum_{m=1}^{M} \left[\nabla_{\phi,\theta,\Pi} \mathcal{J}^{(t)}(\mathbf{x}^{(m,l)}, \mathbf{z}^{(l)}, \mathbf{y}, \phi, \theta, \Pi) \right].$$

2.2.3 Framework of mDPM

We now consider an extension of the partially variational EM framework, where the continuous latent component is modeled via a diffusion process. For a single observation $\mathbf{y}$, we introduce a discrete latent variable $\mathbf{x} \in \{1, \ldots, K\}$ representing the class label, and a continuous latent trajectory $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)$, where each $\mathbf{z}_t \in \mathbb{R}^d$ denotes the latent variable at time step t in the diffusion process.

We distinguish between two sets of model parameters: θ governs the diffusion process and decoder, while Π controls the categorical prior over $\mathbf{x}$. The joint distribution over the latent variables and observation is given by:

$$p(\mathbf{x}, \mathbf{z}, \mathbf{y} \mid \theta, \Pi) = p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}_1, \theta) \left[\prod_{t=1}^{T-1} p(\mathbf{z}_t \mid \mathbf{x}, \mathbf{z}_{t+1}, \theta) \right] p(\mathbf{z}_T) p(\mathbf{x} \mid \Pi),$$

where $p(\mathbf{z}_T)$ is a standard Gaussian prior, $p(\mathbf{x} \mid \Pi)$ is a categorical distribution over the K classes, and the transition probabilities $p(\mathbf{z}_t \mid \mathbf{x}, \mathbf{z}_{t+1}, \theta)$ define the learned reverse diffusion process. The term $p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}_1, \theta)$ denotes the final decoder. We adopt a partially variational inference strategy, in which the variational posterior over the latent variables is factorized as:

$$q(\mathbf{x}, \mathbf{z} \mid \mathbf{y}, \theta^{(l)}, \Pi^{(l)}) = q(\mathbf{z} \mid \mathbf{y}) p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(l)}, \Pi^{(l)}),$$

where $q(\mathbf{z} \mid \mathbf{y})$ represents the fixed forward diffusion process. Since this is a predefined Markov chain determined entirely by the noise schedule, it does not involve any learnable parameters. Therefore, we omit the variational parameters (e.g., ϕ) from the notation.

The forward process factorizes as:

$$q(\mathbf{z} \mid \mathbf{y}) = q(\mathbf{z}_1 \mid \mathbf{y}) \prod_{t=1}^{T-1} q(\mathbf{z}_{t+1} \mid \mathbf{z}_t),$$

which reflects the standard structure of the diffusion model.

The evidence lower bound (ELBO) at iteration l is given by:

$$\begin{aligned}
\mathcal{L}^{(l)}(\theta, \Pi) &= \mathbb{E}_{q(\mathbf{x},\mathbf{z}|\mathbf{y},\theta^{(l)},\Pi^{(l)})}\left[\log \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{y} \mid \theta, \Pi)}{q(\mathbf{x}, \mathbf{z} \mid \mathbf{y}, \theta^{(l)}, \Pi^{(l)})}\right] \\
&= \mathbb{E}_{q(\mathbf{x},\mathbf{z}|\mathbf{y},\theta^{(l)},\Pi^{(l)})}\left[\log \frac{p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}_1, \theta)\prod_{t=1}^{T-1} p(\mathbf{z}_t \mid \mathbf{x}, \mathbf{z}_{t+1}, \theta)p(\mathbf{z}_T)p(\mathbf{x} \mid \Pi)}{q(\mathbf{z} \mid \mathbf{y})p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(l)}, \Pi^{(l)})}\right] \\
&= \mathbb{E}_{\substack{\mathbf{z}\sim q(\mathbf{z}|\mathbf{y}) \\ \mathbf{x}\sim p(\mathbf{x}|\mathbf{z},\mathbf{y},\theta^{(l)},\Pi^{(l)})}}\left[\log \frac{p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}_1, \theta)\prod_{t=1}^{T-1} p(\mathbf{z}_t \mid \mathbf{x}, \mathbf{z}_{t+1}, \theta)p(\mathbf{z}_T)}{q(\mathbf{z} \mid \mathbf{y})} + \log \frac{p(\mathbf{x} \mid \Pi)}{p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(l)}, \Pi^{(l)})}\right] \\
&\propto \mathbb{E}_{\substack{\mathbf{z}\sim q(\mathbf{z}|\mathbf{y}) \\ \mathbf{x}\sim p(\mathbf{x}|\mathbf{z},\mathbf{y},\theta^{(l)},\Pi^{(l)})}}\left[\log \frac{p(\mathbf{y} \mid \mathbf{x}, \mathbf{z}_1, \theta)\prod_{t=1}^{T-1} p(\mathbf{z}_t \mid \mathbf{x}, \mathbf{z}_{t+1}, \theta)p(\mathbf{z}_T)}{q(\mathbf{z} \mid \mathbf{y})} + \log p(\mathbf{x} \mid \Pi)\right] \\
&= \mathcal{L}^{(l)}_{\text{diffusion}}(\theta) + \mathcal{L}^{(l)}_{\text{label}}(\Pi),
\end{aligned}$$

where the first term corresponds to the latent diffusion component and the second term captures the contribution from label inference. A key challenge in this formulation is that the label variable $\mathbf{x}$ is also latent, and must therefore be sampled from its posterior distribution during training.

The posterior distribution over the discrete label variable $\mathbf{x}$ is given by Bayes' rule:

$$\begin{aligned}
p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \theta^{(l)}, \Pi^{(l)}) &\propto p(\mathbf{z}, \mathbf{y} \mid \mathbf{x}, \theta^{(l)})p(\mathbf{x} \mid \Pi^{(l)}), \\
&= \pi_k^{(l)} p(\mathbf{y} \mid \mathbf{x} = k, \mathbf{z}_1, \theta^{(l)}) \prod_{t=1}^{T-1} p(\mathbf{z}_t \mid \mathbf{x} = k, \mathbf{z}_{t+1}, \theta^{(l)})p(\mathbf{z}_T),
\end{aligned}$$

where $\pi_k^{(l)}$ denotes the class prior for $\mathbf{x} = k$ under the current estimate $\Pi^{(l)}$.

Since $\mathbf{x}$ takes values in a finite discrete set $\{1, \ldots, K\}$, the posterior can be expressed in closed form using normalized weights:

$$\begin{aligned}
p(\mathbf{x} = k \mid \mathbf{z}, \mathbf{y}, \theta^{(l)}, \Pi^{(l)}) &= \frac{s_k^{(l)}(\mathbf{z})}{\sum_{k'=1}^{K} s_{k'}^{(l)}(\mathbf{z})}, \\
\text{where} \quad s_k^{(l)}(\mathbf{z}) &= \pi_k^{(l)} p(\mathbf{y} \mid \mathbf{x} = k, \mathbf{z}_1, \theta^{(l)}) \prod_{t=1}^{T-1} p(\mathbf{z}_t \mid \mathbf{x} = k, \mathbf{z}_{t+1}, \theta^{(l)})p(\mathbf{z}_T).
\end{aligned}$$

Computing $s_k^{(l)}(\mathbf{z})$ exactly requires evaluating the full latent trajectory $(\mathbf{z}_1, \ldots, \mathbf{z}_T)$, which is computationally expensive due to the multiplicative structure of the diffusion likelihood. To reduce this cost while retaining temporal information beyond the final decoding step, we adopt a log-space multi-step approximation based on randomly sampled timesteps.

Specifically, we begin by expressing the unnormalized class score in the log domain:

$$\log s_k^{(l)}(\mathbf{z}) = \log \pi_k^{(l)} + \log p(\mathbf{y} \mid \mathbf{x} = k, \mathbf{z}_1, \theta^{(l)}) + \sum_{t=1}^{T-1} \log p(\mathbf{z}_t \mid \mathbf{x} = k, \mathbf{z}_{t+1}, \theta^{(l)}) + \log p(\mathbf{z}_T).$$

The summation over timesteps in this expression involves a potentially large number of terms, which can be expensive to compute exactly. To address this, we approximate the summation using Monte Carlo estimation by uniformly sampling a subset of timesteps. Specifically, we sample M timesteps $\{t_m\}_{m=1}^{M}$ independently from the uniform distribution over $\{1, \ldots, T-1\}$, and estimate the sum as:

$$\sum_{t=1}^{T-1} \log p(\mathbf{z}_t \mid \mathbf{x} = k, \mathbf{z}_{t+1}) \approx \frac{T-1}{M} \sum_{m=1}^{M} \log p(\mathbf{z}_{t_m} \mid \mathbf{x} = k, \mathbf{z}_{t_m+1}, \theta^{(l)}).$$

Substituting this estimate into the original expression yields the Monte Carlo approximation of the log-score:

$$\log \tilde{s}_k^{(l)}(\mathbf{z}) := \log \pi_k^{(l)} + \log p(\mathbf{y} \mid \mathbf{x} = k, \mathbf{z}_1, \theta^{(l)}) + \frac{T-1}{M} \sum_{m=1}^{M} \log p(\mathbf{z}_{t_m} \mid \mathbf{x} = k, \mathbf{z}_{t_m+1}, \theta^{(l)}) + \log p(\mathbf{z}_T).$$

Here, each term $p(\mathbf{z}_{t_m} \mid \mathbf{z}_{t_m+1}, \mathbf{x} = k, \theta^{(l)})$ corresponds to a reverse transition in the diffusion process, and is typically modeled as a Gaussian whose mean is parameterized by a denoising network conditioned on the class label k. This denoising prediction is often implemented in terms of noise prediction $\epsilon_{\theta^{(l)}}$, following the DDPM parameterization.

Given the approximate log-scores $\log \tilde{s}_k^{(l)}(\mathbf{z})$, we compute the approximate posterior over class labels using a softmax:

$$\tilde{p}(\mathbf{x} = k \mid \mathbf{z}, \mathbf{y}, \theta^{(l)}, \Pi^{(l)}) = \frac{\exp(\log \tilde{s}_k^{(l)}(\mathbf{z}))}{\sum_{k'=1}^{K} \exp(\log \tilde{s}_{k'}^{(l)}(\mathbf{z}))}.$$

This posterior captures both the conditional likelihood of the observation $\mathbf{y}$ given the latent $\mathbf{z}_1$, and the consistency of the intermediate latent trajectory $\{\mathbf{z}_{t_m}\}$ under a class-conditional reverse process. By drawing samples from $\tilde{p}(\mathbf{x} \mid \mathbf{z}, \mathbf{y})$, we can obtain class labels to condition the standard DDPM training procedure. That is, after sampling $\mathbf{x}$, we proceed with noise prediction training as in the original DDPM framework, using $\mathbf{x}$ as a condition. This allows us to learn a class-conditional generative model without requiring labeled data, leveraging the model itself to infer pseudo-labels in a self-consistent fashion.

2.3 Hidden Markov Models with Deep Generative Emissions

2.3.1 Framework of HMM-VAE

Suppose we have K hidden states, denoted by the set $S = \{s_1, s_2, \ldots, s_K\}$. The hidden state sequence is $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$, and the corresponding observation sequence is $\boldsymbol{Y} = (y_1, y_2, \ldots, y_n)$. To enhance the expressiveness of the emission model, we introduce a sequence of continuous latent variables $\boldsymbol{Z} = (z_1, z_2, \ldots, z_n)$, where each $z_i \in \mathbb{R}^d$ serves as the latent representation in a variational autoencoder (VAE) framework.

The initial state distribution is specified by $\Pi = \{\pi_k\}_{k=1}^K$, where $\pi_k = p(x_1 = s_k \mid \Pi)$, and the transition matrix is $A = \{a_{ij}\}_{1 \le i,j \le K}$, where $a_{ij} = p(x_t = s_j \mid x_{t-1} = s_i, A)$. For each state s_k and time step i, we define the emission probability as

$$b_k(y_i, z_i) := p(y_i, z_i \mid x_i = s_k, \theta) = p(y_i \mid z_i, x_i = s_k, \theta) \cdot p(z_i),$$

where θ denotes the parameters of the decoder, and $p(z_i)$ is typically a standard Gaussian prior.

We adopt a partially variational inference strategy, where the posterior over the continuous latent variables $\boldsymbol{Z}$ is approximated by a variational distribution $q_\phi(z_i \mid y_i)$ with parameters ϕ, and the posterior over the discrete state sequence $\boldsymbol{X}$ is inferred exactly via the conditional distribution. The joint approximate posterior is therefore given by

$$q(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)}, \phi) = p(\boldsymbol{X} \mid \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)}) \cdot \prod_{i=1}^{n} q_\phi(z_i \mid y_i),$$

where $\theta^{(l)}, \Pi^{(l)}, A^{(l)}$ are the current estimates of the model parameters.

Due to the Markovian structure of the HMM, the joint distribution of all variables factorizes as

$$p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{Y} \mid \theta, \Pi, A) = p(x_1 \mid \Pi) \cdot p(y_1, z_1 \mid x_1, \theta) \cdot \prod_{i=2}^{n} p(x_i \mid x_{i-1}, A) \cdot p(y_i, z_i \mid x_i, \theta).$$

The evidence lower bound (ELBO) at iteration l is defined as

$$\mathcal{L}^{(l)}(\theta, \phi, \Pi, A) = \mathbb{E}_{q(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)}, \phi)} \left[\log \frac{p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{Y} \mid \theta, \Pi, A)}{q(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)}, \phi)} \right].$$

Substituting the expressions for the joint and approximate posteriors, we obtain

$$\mathcal{L}^{(l)}(\theta, \phi, \Pi, A) = \mathbb{E}_q \left[\sum_{i=1}^{n} \log \frac{p(y_i, z_i \mid x_i, \theta)}{q_\phi(z_i \mid y_i)} + \log \frac{p(x_1 \mid \Pi) \cdot \prod_{i=2}^{n} p(x_i \mid x_{i-1}, A)}{p(\boldsymbol{X} \mid \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)})} \right].$$

This expression can be interpreted as taking the expectation first over the variational posterior of $\boldsymbol{Z}$, and then over the exact conditional posterior of $\boldsymbol{X}$:

$$\mathcal{L}^{(l)}(\theta, \phi, \Pi, A) \propto \mathbb{E}_{\substack{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\boldsymbol{Y}) \\ \boldsymbol{X}\sim p(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{Y},\theta^{(l)},\Pi^{(l)},A^{(l)})}} \left[\sum_{i=1}^{n} \log \frac{p(y_i, z_i \mid x_i, \theta)}{q_\phi(z_i \mid y_i)} + \log p(x_1 \mid \Pi) + \sum_{i=2}^{n} \log p(x_i \mid x_{i-1}, A) \right].$$

We denote the ELBO as the sum of two components:

$$\mathcal{L}^{(l)}(\theta, \phi, \Pi, A) = \mathcal{L}^{(l)}_{\text{emission}}(\theta, \phi) + \mathcal{L}^{(l)}_{\text{transition}}(\Pi, A),$$

where the first term corresponds to the VAE-based emission modeling, and the second term reflects the structured transition modeling of the HMM.

Unlike mixture models, HMM-based models require a more structured sampling procedure due to the Markov dependencies among the latent variables $\boldsymbol{X}$. Specifically, to sample $\boldsymbol{X} \sim p(\boldsymbol{X} \mid \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)})$, we adopt a forward-backward sampling strategy analogous to that used in classical HMMs.

We begin by defining the forward variable at iteration l as

$$\alpha_k^{(l)}(i) := p\left(y_{1:i}, z_{1:i}, x_i = s_k \mid \theta^{(l)}, \Pi^{(l)}, A^{(l)}\right),$$

where $y_{1:i} = (y_1, \ldots, y_i)$, $z_{1:i} = (z_1, \ldots, z_i)$, and $x_i = s_k$ denotes the latent state at time i.

The forward recursion is initialized by

$$\alpha_k^{(l)}(1) = p(x_1 = s_k \mid \Pi^{(l)}) \cdot p(y_1, z_1 \mid x_1 = s_k, \theta^{(l)}) = \pi_k^{(l)} \cdot b_k^{(l)}(y_1, z_1),$$

where

$$b_k^{(l)}(y_i, z_i) := p(y_i, z_i \mid x_i = s_k, \theta^{(l)}) = p(y_i \mid x_i = s_k, z_i, \theta^{(l)}) \cdot p(z_i).$$

For $i \geq 2$, the forward recursion proceeds as

$$\begin{aligned}
\alpha_k^{(l)}(i) &= p(y_{1:i}, z_{1:i}, x_i = s_k \mid \theta^{(l)}, \Pi^{(l)}, A^{(l)}) \\
&= p(y_i, z_i \mid x_i = s_k, \theta^{(l)}) \cdot \sum_{j=1}^{K} \alpha_j^{(l)}(i-1) \cdot a_{jk}^{(l)} \\
&= b_k^{(l)}(y_i, z_i) \cdot \sum_{j=1}^{K} \alpha_j^{(l)}(i-1) a_{jk}^{(l)}.
\end{aligned}$$

The forward computation proceeds sequentially:

$$\{\alpha_k^{(l)}(1)\}_k \to \{\alpha_k^{(l)}(2)\}_k \to \cdots \to \{\alpha_k^{(l)}(n)\}_k, \quad \text{for } k = 1, \ldots, K.$$

Once the forward pass is completed, we perform backward sampling to obtain a sample from the posterior over the latent state sequence.

We start by sampling x_n from its marginal posterior:

$$p(x_n = s_k \mid \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)}) \propto \alpha_k^{(l)}(n),$$

which is normalized as

$$p(x_n = s_k \mid \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)}) = \frac{\alpha_k^{(l)}(n)}{\sum_{k'=1}^{K} \alpha_{k'}^{(l)}(n)}.$$

Then, for $i = n-1, n-2, \ldots, 1$, we sample backward from the conditional distribution:

$$\begin{aligned} p(x_i = s_k \mid x_{i+1} = s_j, \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)}) &\propto p(y_{1:i}, z_{1:i}, x_i = s_k \mid \theta^{(l)}, \Pi^{(l)}, A^{(l)}) \cdot p(x_{i+1} = s_j \mid x_i = s_k, A^{(l)}) \\ &= \alpha_k^{(l)}(i) \cdot a_{kj}^{(l)}. \end{aligned}$$

The normalized sampling probability is given by

$$p(x_i = s_k \mid x_{i+1} = s_j, \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}, \Pi^{(l)}, A^{(l)}) = \frac{\alpha_k^{(l)}(i) \cdot a_{kj}^{(l)}}{\sum_{k'=1}^{K} \alpha_{k'}^{(l)}(i) \cdot a_{k'j}^{(l)}}.$$

Thus, the full backward sampling procedure proceeds as

$$x_n \to x_{n-1} \to \cdots \to x_1.$$

Then we can train the VAE after sampling X using Gumbel softmax mentioned above.

After training, we can predict the hidden states by the Viterbi algorithm. It is similar to the backward sampling, but a little different. The backward sampling is sampling, the viterbi algorithm is finding the maximum.

$$x_n^* = \arg\max_k p(x_n = s_k \mid \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}) = \arg\max_k \alpha_k(n)$$

Then sample $i = n-1, n-2, \cdots, 1$:

$$x_i^* = \arg\max_k p(x_i = s_k \mid x_{i+1} = s_j, \boldsymbol{Z}, \boldsymbol{Y}, \theta^{(l)}) = \arg\max_k \alpha_k(i) a_{kj}$$

where $\alpha_k(i)$ is approximated by $\tilde{\alpha}_k(i)$.

2.3.2 Framework of HMM-DPM

The difference is: In diffusion, it is complex to calculate $b_k^{(l)}(y_i, z_i)$:

$$\begin{aligned} b_k^{(l)}(y_i, z_i) &= p_{\theta^{(l)}}(y_i, z_i \mid x_i = s_k) \\ &= p_{\theta^{(l)}}(y_i \mid x_i, z_i) \prod_{t=1}^{T-1} p_{\theta^{(l)}}(z_{i,t} \mid x_i, z_{i,t+1}) p(z_{i,T}) \end{aligned}$$

We do similar procedure in mixture of diffusion to estimate ....

And we don't have ϕ, we don't need reparameterization trick.

3 Results

3.1 Evaluation on Mixture Models

3.2 Evaluation on HMMs

4 Discussion

we provide end-to-end algorithm for ..

By doing so, it allows us to leverage the high representational capacity of deep generative models for modeling complex data, while preserving interpretability and analytical tractability for a subset of model components.