Auto-Encoding Variational Bayes

Chen Yankun[1*], *LiuJingxuan*[2], *PengLingyun*[3], *WuYiqi*[4], *XuYige*[5], *ZhangZhanhao*[6]

January 3, 2024

1 Bell Honors School, Nanjing University of Posts and Telecommunications, Nanjing 210037, China
2 International Elite Engineering School, East China University of Science and Technology, Shanghai 200237, China
3 College of Cyberspace Security, Guangzhou University, Guangzhou 510006, China
4 International college of engineering, Changsha University of Science and Technology, Changsha,410000, China
5 College of Electronic Science and Engineering, Jilin University, Changchun, 130012, China
6 School of Mechanical Transport And Engineering, Chongqing University, Chongqing 400030, China
* Correspondence:q20010211@njupt.edu.cn;Tel:+86-18994118171

Abstract

This paper employs the Auto-Encoding Variational Bayes (AEVB) estimator based on Stochastic Gradient Variational Bayes (SGVB), designed to optimize recognition models for challenging posterior distributions and large-scale datasets. It has been applied to the mnist dataset and extended to form a Dynamic Bayesian Network (DBN) in the context of time series. The paper delves into Bayesian inference, variational methods, and the fusion of Variational Autoencoders (VAEs) and variational techniques. Emphasis is placed on reparameterization for achieving efficient optimization. AEVB employs VAEs as an approximation for intricate posterior distributions.

1 Introduction

We discuss the comparison of different algorithms for approximate inference with continuous latent variables, such as the wake-sleep algorithm, Monte Carlo EM, and Variational Bayes with Auto-Encoding Variational Bayes (AEVB)[1]. The AEVB algorithm is introduced as a new estimator for the variational lower bound to achieve efficient approximate inference. The paper also mentions the use of a learning encoder to project high-dimensional data onto a low-dimensional manifold. Future research directions are discussed, including learning hierarchical generative architectures, time series models[2], and supervised models with latent variables.

2 Background

The core of probabilistic modeling is Bayesian inference, a pivotal principle guiding parameter estimation through the fusion of observed data and prior beliefs based on Bayes' theorem. Variational inference complements this by handling intricate scenarios where computing the exact posterior is infeasible. It approximates the true posterior using a simpler distribution, optimizing divergence measures.

Generating models simulate the inherent data generation process. These models enable the creation of synthetic data samples that align with the characteristics of the observed dataset, proving beneficial for tasks such as data augmentation and anomaly detection. Concurrently, autoencoders, functioning as neural network architectures, delve into the realm of data compression. They consist of an encoder component that condenses data into a lower-dimensional latent space, along with a decoder component that reconstructs the initial data using this latent representation.

Variational autoencoders (VAEs) amalgamate these concepts, merging the principles of autoencoders and variational inference. They formulate an encoder that transforms data into a latent space characterized by probability distributions, alongside a decoder that generates novel data instances. VAEs leverage the reparameterization technique, which streamlines the computation of gradients and the optimization of the model.

AEVB leverages the VAE framework to approximate intricate posterior distributions within generative models. It seamlessly integrates variational inference into the generative model, harnessing neural networks for data encoding and decoding. The algorithm finds synergy with stochastic gradient optimization, maximizing the variational lower bound and accelerating the training process.

At the core of AEVB's optimization lies the Stochastic Gradient Variational Bayes (SGVB) method. SGVB employs stochastic gradient descent to update parameters within the recognition model, thereby enhancing scalability. Through the utilization of data mini-batches, SGVB is adept at managing substantial datasets and intricate models[5].

3 Methods and Results sections

The approach outlined in this section can be used to obtain a lower bound estimate, which is a stochastic objective function, for a range of directed graphical models that have continuous latent variables. We are limiting our focus to the prevalent scenario where we use an i.i.d. dataset, where each datapoint is associated with its own latent variables. Our objective is to conduct maximum likelihood or maximum a posteriore estimation of the global parameters while employing variational inference techniques on the latent variables.

3.1 Methods

In this section the thesis cited gave an example where they use a neural network for the probabilistic encoder $q_{\boldsymbol{\phi}(\mathbf{z}|\mathbf{x})}$ (the approximation to the posterior of the generative model $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ and where the parameters ϕ and θ are optimized jointly with the AEVB algorithm.

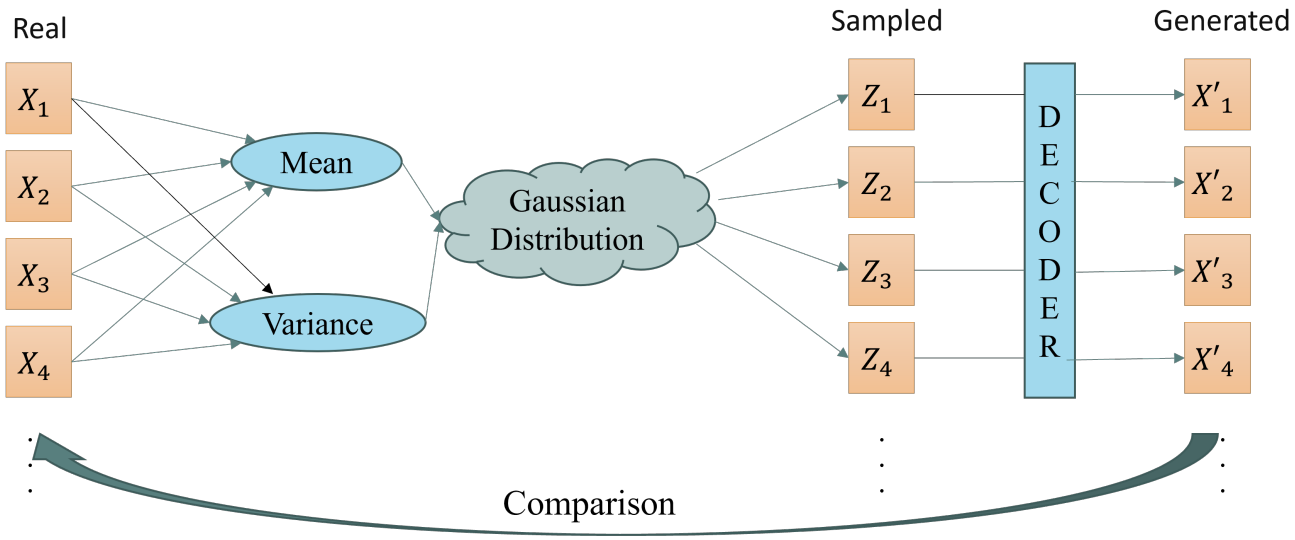

Figure 1: Directed Graphical Model of AEVB

In this context, $q_\phi(z|x)$ can be considered as a probabilistic encoder, and $p_\theta(x|z)$ can be considered as a probabilistic decoder. The real samples x are encoded into z and a new sample x, similar to x, is generated. Next, the real samples x are compared with the generated samples x. Their loss, KL-divergence, and likelihood are then calculated[3].

The marginal likelihood is composed of a sum over the marginal likelihoods of individual data points, which can each be written using the given equation.

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) \tag{1}$$

On the right side, the first term represents the KL divergence between the approximate and true posterior, which is non-negative. As a result, the second term is referred to as the lower bound on the marginal likelihood of datapoint i and can be written as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = E_{q\boldsymbol{\phi}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] \tag{2}$$

The reparameterization trick is a technique used in the Auto-Encoding Variational Bayes (AEVB) algorithm, which is an efficient way to learn deep generative models. The trick is used to estimate the gradient of the variational lower bound, which is intractable in general. It allows the model to be trained efficiently using backpropagation[6].

In AEVB, the reparameterization trick works by introducing an auxiliary noise variable with a simple distribution, such as a standard normal distribution. The latent variable z is then expressed as a deterministic function of the noise variable and the variational parameters: $z = g_\phi(\epsilon, x)$, where g_ϕ is a differentiable function. This reparameterization allows us to estimate the gradient of the variational lower bound[3] using Monte Carlo methods and backpropagate through the sampling process.

The reparameterization trick is crucial for implementing the AEVB algorithm and has been widely used in various deep generative models. It enables efficient optimization of the variational lower bound using standard stochastic gradient methods.

Let the prior over the latent variables be the centered isotropic multivariate Gaussian $p_{\boldsymbol{\theta}}(\mathbf{z}) = N(z; 0, I)$ Note that in this case, the prior lacks parameters. We let $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ be a multivariate Gaussian (in case of real-valued data) or Bernoulli (in case of binary data) whose distribution parameters are computed from z with an MLP.Note the true

posterior $p_{\boldsymbol{\theta}}(\mathbf{z}) = N(z; 0, I)$ is in this case intractable. While there is much freedom in the form $q_{\boldsymbol{\phi}(\mathbf{z}|\mathbf{x})}$, we will assume the true (but intractable) posterior takes on an approximate Gaussian form with an approximately diagonal covariance. In this case, we can let the variational approximate posterior be a multivariate Gaussian with a diagonal covariance structure[4].

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I}) \tag{3}$$

where the mean and s.d. of the approximate posterior, $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{(i)}$, are outputs of the encoding MLP, i.e. nonlinear functions of datapoint $\boldsymbol{x}^{(i)}$ and the variational parameters ϕ Then we sample from the posterior z using $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)}$ and $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$.We signify an element-wise product. In this model both the prior and $q_{\boldsymbol{\phi}(\mathbf{z}|\mathbf{x})}$ are Gaussian; in this case, we can use the estimator of eq.(7) where the KL divergence can be computed and differentiated without estimation. The resulting estimator for this model and datapoint $x^{(i)}$ is:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) &\simeq \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}) \\ &\text{where} \quad \mathrm{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})\end{aligned} \tag{4}$$

where the decoding term $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z^{(i,l)})$is a Bernoulli or Gaussian MLP, depending on the type of data we are modeling.

3.2 Code

Within this section, we furnish the code implementation for the Variational Autoencoder (VAE) employing the AEVB training algorithm. The code structure is divided across two distinct files: "main.py" and "model.py".This implementation is built using PyTorch and utilizes the MNIST dataset for training.

The 'model.py' file contains the definition of the 'VariationalAutoencoder' class, which inherits from 'nn.Module'. This class encapsulates the architecture and training logic of the VAE.

The constructor initializes the VAE's architecture, including the encoder and decoder neural networks, and defines the optimizer for parameter updates during training.

Methods are defined for encoding, decoding, and reparameterizing latent variables. The forward pass combines these steps to reconstruct an image from input data.

The 'train' method implements the training logic. It computes the reconstruction loss using mean squared error (MSE) and the Kullback-Leibler (KL) divergence loss, which ensures that the latent variables follow a desired distribution. The total loss is a combination of these two components, and backpropagation is used to update the model's parameters.

The 'main.py' file is responsible for setting up the training loop and testing the VAE. It starts by importing necessary libraries and defining command-line arguments using the 'argparse' library. These arguments include the batch size, number of epochs, hidden dimension, and latent space dimensionality. The code demonstrates the loading of the MNIST dataset, transformation, and data loading using PyTorch's 'DataLoader'.

The 'VariationalAutoencoder' class in 'model.py' is instantiated, and the model is moved to the desired device. The training loop begins by iterating through epochs, and within each epoch, iterating through batches of training data. The model's 'train' method is called to update its parameters based on reconstruction and regularization losses.

After each epoch, a testing loop is executed. The VAE is used to reconstruct a given image and generate new images from random latent variables. The reconstructed and generated images are saved for visualization, which will be shown in the result part.

3.3 Results

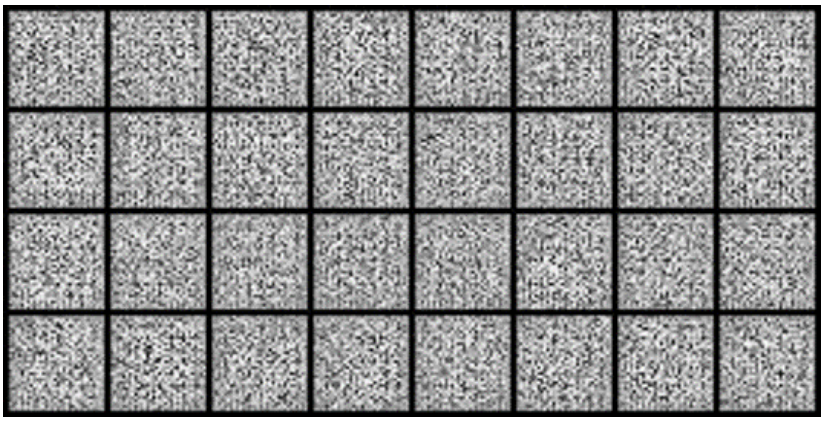

Figure 2: Failed Result of the MNIST Dataset

The root cause of the initial error was the utilization of antiquated code that was incompatible with the employed version of PyTorch. The libraries in use had undergone numerous modifications, rendering the project library, referenced within the code, absent in the contemporary version, culminating in an unsuccessful output.

Figure 3: Successful Reconstructed Results of the MNIST Dataset

The initial four lines represent the manually provided handwritten data, while the subsequent four lines depict a reconstructed outcome derived from the original image. In this scenario, the latent variables are generated through the encoding of the input image. Subsequently, the reconstructed image is produced by decoding this latent variable. This process accounts for the close resemblance between the resultant output and the original image.

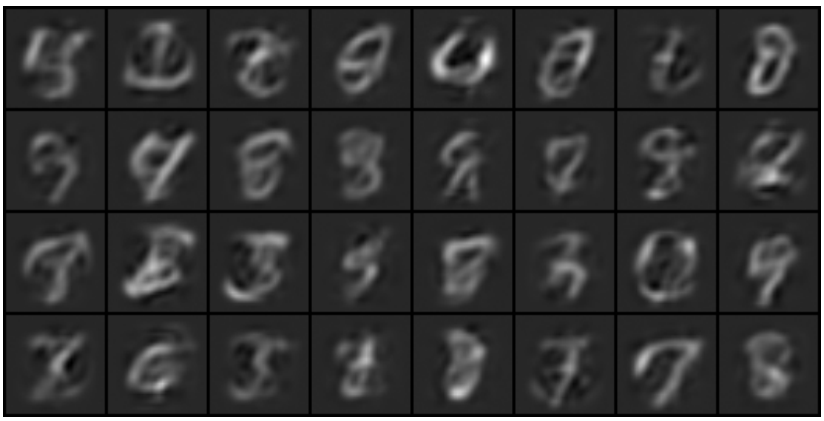

Figure 4: Successful Generated Results of the MNIST Dataset

This figure displays the outcomes generated by utilizing randomly selected latent variables. The generated images might appear slightly blurry due to the nature of the latent variables provided.

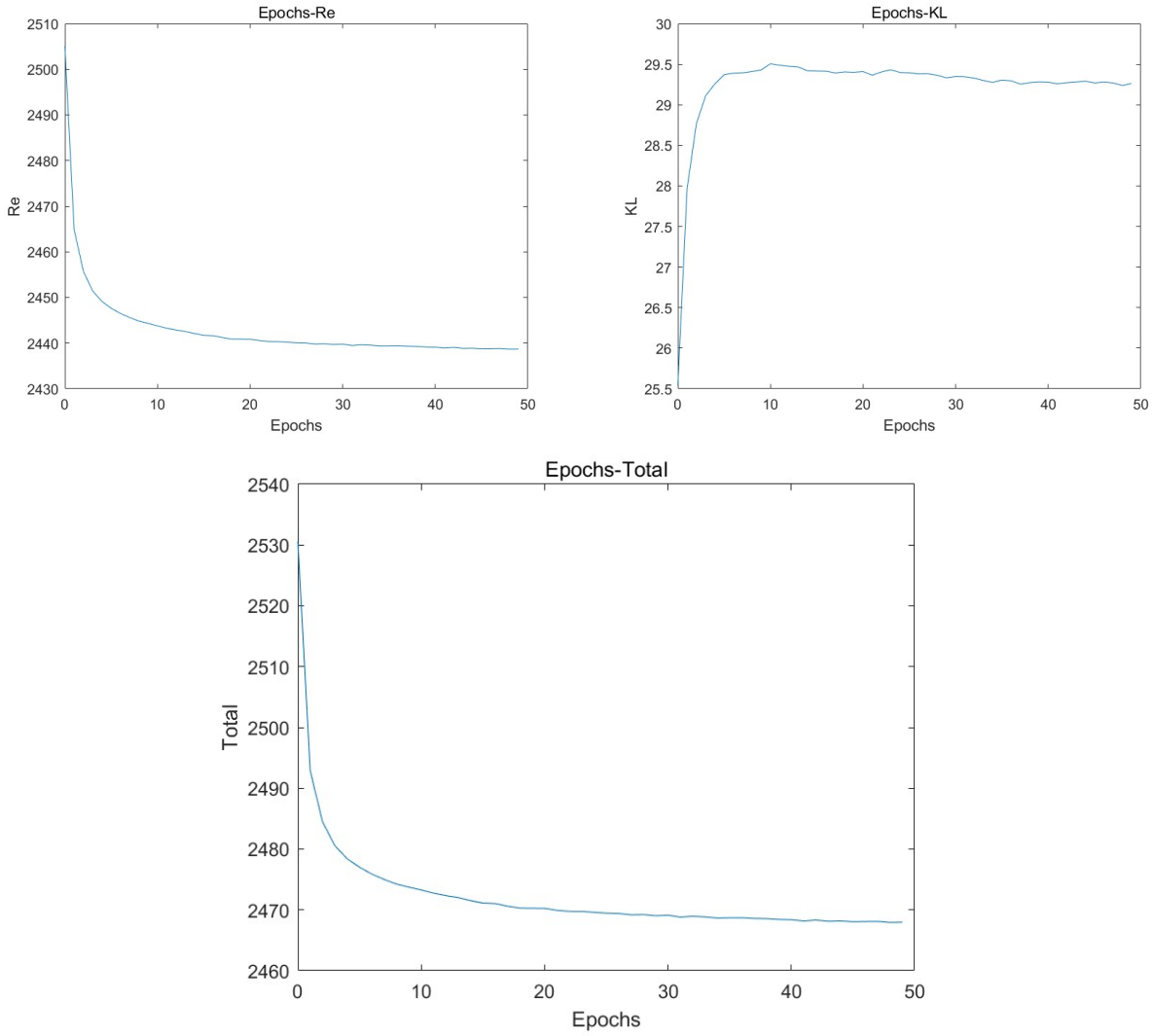

Figure 5: BCE loss, KL divergence, and total loss trends over 50 epochs

The three figures above show the reconstruction loss (BCE loss), dropped from 2500 to around 2440. This indicates that the model learned to reconstruct input data more accurately as training progressed. The KL-divergence, however, rose from 25.5 to 29.5 after training and converged. This trend is expected in VAEs, as the model aims to adjust its latent variable distribution to match the desired distribution. The strategy is to sacrifice KL-divergence in favor of improving reconstruction. Finally, The total loss function dropped from around 2530 to around 2470 and converged.

This trend aligns with the patterns seen in both the reconstruction and KL divergence losses.

To summarize, the training procedure involves prioritizing the harmony between the reconstruction loss and KL-divergence, with the goal of minimizing the total loss.

4 Extension

4.1 Hidden Markov Model

The Hidden Markov Model is a probabilistic model about sequences. It describes the process where an underlying Markov chain generates an unobservable sequence of states, followed by the generation of an observed sequence of outcomes based on each state[7].

After comprehending and successfully reproducing Auto-Encoding Variational Bayesian, we try to apply it to the domain of time series, for example, Hidden Markov Model (HMM). This network is derived from a state with the same structure, unfolding along the time axis. It is impossible to know the state directly and we often use the observed value to infer the state. The Hidden Markov Model is made from Transition Probabilities, State and Observed value.

The Hidden Markov Model will be described as follows: $\lambda = (A, B, \pi)$ The λ is the model itself. The A means State Transition Probability Distribution and B represents Observed Value Transition Probability Distribution and π represents Initial Probability Distibution. They are all matrixes.

When we want to generate an observation sequence, the first step is generating state i_1 and recording it to t_1. Then it generates observed value o_1 through Observed Value Transition Probability Distribution according to i_1. Later it generates the next state i_2 according to State Transition Probability Distribution and generates o_2. Then repeat the steps above until the end.

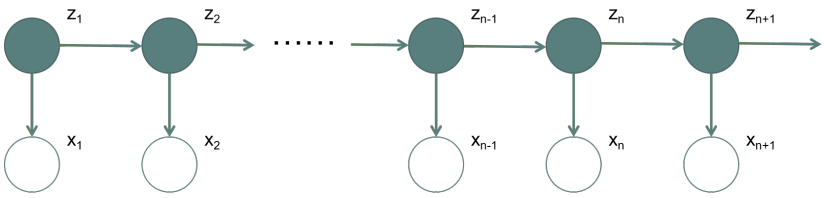

Figure 6: The Hidden Markov Model

4.2 Main problems of the Hidden Markov Model

Evaluation Problem We want to calculate the probability of a given observation sequence. This aims to evaluate whether our model is proper or not. We use λ and Observation Sequence as input and get the conditional probability as output. The forward algorithm can be used.

Learning Problem We want to learn parameters from the observation sequence, and we get the proper parameters by MLE or MAP. We use Observation Sequence as input and we want to get the parameters of λ.

Decoding Problem We want to infer the most possible state sequence according to the observation sequence. This is the most important part. We believe it is correct to use Auto-Encoding Variational Bayesian here to encode x, infer the probability distribution z, and decode the hidden variable to the observed x'.

4.3 Relationship between Hidden Markov Model and AEVB

When we extend it on the time axis, we get a structure that is similar to the Hidden Markov Model. This provides significant potential for advancing time-related issues. If possible, we aspire to enhance the existing Hidden Markov Model by leveraging the strengths of both methodologies, leading to substantial improvements in accuracy and efficiency. This enhancement aims to elevate the overall performance of our approach in the realm of time series modeling, e.g. classification of behaviors in a video.

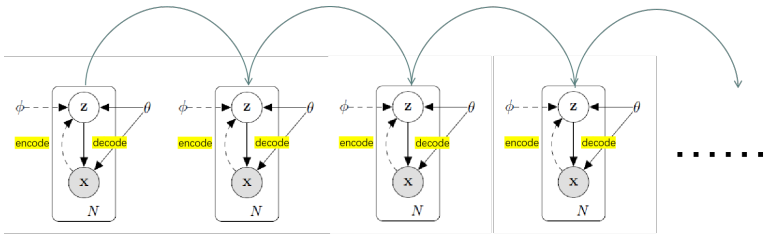

Figure 7: extend one single AEVB to time axis

5 Conclusion and future directions

5.1 Conclusion

A new variational lower bound estimator, stochastic gradient VB (SGVB), is introduced in the case of efficient approximate inference of continuous latent variables. Using the standard stochastic gradient method, the proposed estimator can be directly discretized and optimized. Also for the i.i.d. datasets and continuous latent variables for each data point, an efficient algorithm for efficient inference and learning is introduced, the Auto-Encoding VB (AEVB), which learns an approximate inference model using the SGVB estimator. We use this algorithm for code reproduction of the literature content.

5.2 Future Directions

This has promising potential for advancing action classification tasks but also provides a valuable avenue for addressing other time-dependent problems. If possible, by leveraging the strengths of both methodologies, we anticipate improving the existing dynamic Bayesian network and achieving substantial improvements in accuracy and efficiency, thereby enhancing the overall performance of our approach in time-series modeling and classification endeavors.

(i) learning hierarchical generative architectures with deep neural networks (e.g. convolutional networks) used for the encoders and decoders, trained jointly with AEVB;
(ii) time-series models (i.e. dynamic Bayesian networks);
(iii) application of SGVB to the global parameters;
(iv) supervised models with latent variables, useful for learning complicated noise distributions

Author Contributions

Conceptualization: Y.C.,J.L.,L.P.,Y.W.,Y.X. and Z.Z.; methodology,formal analysis: Y.X. and J.L.; code, resources, dataset: J.L. and Y.C.; data analysis, visualization: Z.Z. and L.P.; Extension and future work: Y.C. and Y.W.; writing - original draft preparation: L.P.,Y.W. and Z.Z; writing - reviewing, editing, and supervision: Y.C., J.L., and Y.X.; Project administration: Y.W. and Y.C. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Research Guidelines

This study followed the research guidelines of the Advanced Machine Learning & Machine Intelligence, 2023 Cambridge Summer Academic Programme.

Informed Consent Statement

Informed consent was obtained from all subjects involved in the study.

Data Availability

Please contact the corresponding author(s) for all reasonable requests for access to the data.

Acknowledgments

We would like to give our thanks to our faculty professors and teachers at Cambridge University for their guidance.

Conflicts of Interest

The authors declare no conflict of interest.

Intellectual Property

The authors attest that copyright belongs to them, the article has not been published elsewhere, and there is no infringement of any intellectual property rights as far as they are aware.

References

[1] Kingma, D.P., Welling, M. (2013). *Auto-encoding variational bayes.* arXiv preprint arXiv:1312.6114.

[2] Luo, Y., Wu, T.D., Hwang, J.N. (2003). *Object-based analysis and interpretation of human motion in sports video sequences by dynamic Bayesian networks.* Computer Vision and Image Understanding, 92(2-3), 196-216.

[3] Lopez, R., Regier, J., Jordan, M.I., Yosef, N. (2018). *Information constraints on auto-encoding variational Bayes.* Advances in neural information processing systems, 31.

[4] Lopez, R., Boyeau, P., Yosef, N., Jordan, M., Regier, J. (2020). *Decision-making with auto-encoding variational Bayes.* Advances in Neural Information Processing Systems, 33, 5081-5090.

[5] Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR).*

[6] Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML).*

[7] Rabiner, Lawrence R *A tutorial on hidden Markov models and selected applications in speech recognition* Proceedings of the IEEE