Auto Encoding Variational Bayes

BY DIEDERIK P. KINGMA AND MAX WELLING

Introduction

How to efficiently learn the parameters of directed probabilistic models whose continuous latent variables have intractable posterior distributions? The variational approach to approximate Bayesian inference involves the introduction of an approximate posterior to the intractable posterior, used to maximize the variational lower bound of the marginal likelihood. In the paper they show how to yield a novel and practical estimator of the variational lower bound that can be differentiated and jointly optimized using standard stochastic gradient ascent techniques.

They got this using reparameterization of the expectation of the approximate posterior. All parameters updates, including those of the noise distribution, correspond to optimization of the variational lower bound of the marginal likelihood.

Introduction

The probabilistic encoder can be used for fast approximate inference of latent variables, i.e. for recognition, representation or visualization purposes. Furthermore, the lower bound estimator can be used for unsupervised inference tasks such as denoising and inpainting.

Method

They derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables.

Inputs: dataset with latent variables per datapoint.

Goal: They perform ML or MAP inference on the (global) parameters, and variational inference on the latent variables.

Problem scenario

Consider some dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ consisting of *N* samples of some continuous or discrete variable **x**.

The data are generated by some random process, involving an unobserved continuous random variable **z**. The process consists of two steps:

1) a value **z** (*i*) is generated from some prior distribution $p_{\boldsymbol{\theta}^*}(\mathbf{z})$

2) a value **x** (*i*) is generated from some conditional distribution $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$

They are interested in a general algorithm that even works in the case of:

A) Intractability : the case where the integral of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, d\mathbf{z}$

Is intractable.

B) A large dataset: they have so much data that batch optimization is too costly; they would like to make parameter updates using small minibatches or even single datapoints.

Problem scenario

They propose a solution to three related problems in the above scenario:

1. Efficient approximate maximum likelihood (ML) or maximum a posteriori (MAP) estimation for the parameters θ. The parameters can be of interest themselves. They also allow us to mimic the hidden random process and generate artificial data that resembles the real data.

2. Efficient approximate posterior inference of the latent variable **z** given an observed value **x** for a choice of parameters θ. This is useful for coding or data representation tasks.

3. Efficient approximate marginal inference of the variable **x**. This allows us to perform all kinds of inference tasks where a prior over **x** is required. Common applications in computer vision include image denoising, inpainting and super-resolution.

Problem scenario

For the purpose of solving the above problems, they introduce: the parametric variational approximation $q_{\phi}(\mathbf{z}|\mathbf{x})$ (*(variational) encoder)* : an approximation to the intractable true posterior $p_{\theta}(\mathbf{z}|\mathbf{x})$ (generative decoder).

The variational bound

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints, which can each be rewritten as:

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

The first RHS term is the KL divergence of the approximate from the true posterior, which is non negative. The second RHS term denotes the variational lower bound of the marginal likelihood of datapoint *i*:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right) d\mathbf{z} \quad (2)$$

They would like to optimize the lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$(eq. (2)) using stochastic gradients. Note that following these gradients would either decrease the KL divergence between the approximate and true posterior distributions, or increase the marginal likelihood, or both.

Estimation of the lower bound

The authors introduced a practical estimator and its derivative with respect to the parameters. Recognition model parameters ϕ and generative model parameters θ.

The first term is the KL divergence, can then be interpreted as regularizing ϕ.

The second term is an expected negative reconstruction error.

$$\widetilde{\mathcal{L}}^{B}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

The SGVB estimator and AEVB algorithm

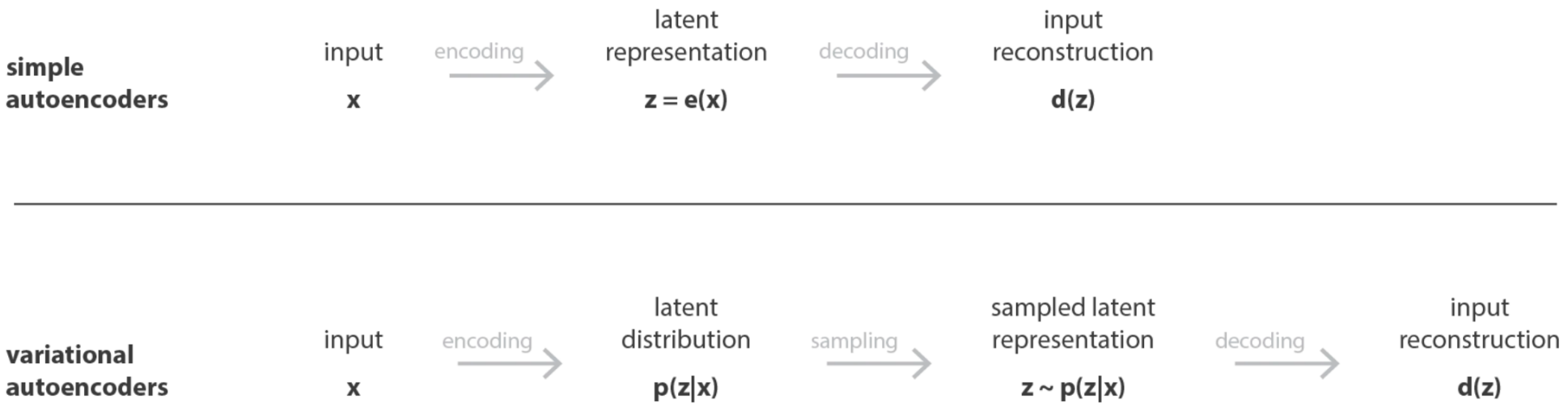
Uses reparameterizated variable z and differentiable transformation gϕ(ε, x) with auxillary variable ε.

$$\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)} | \mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

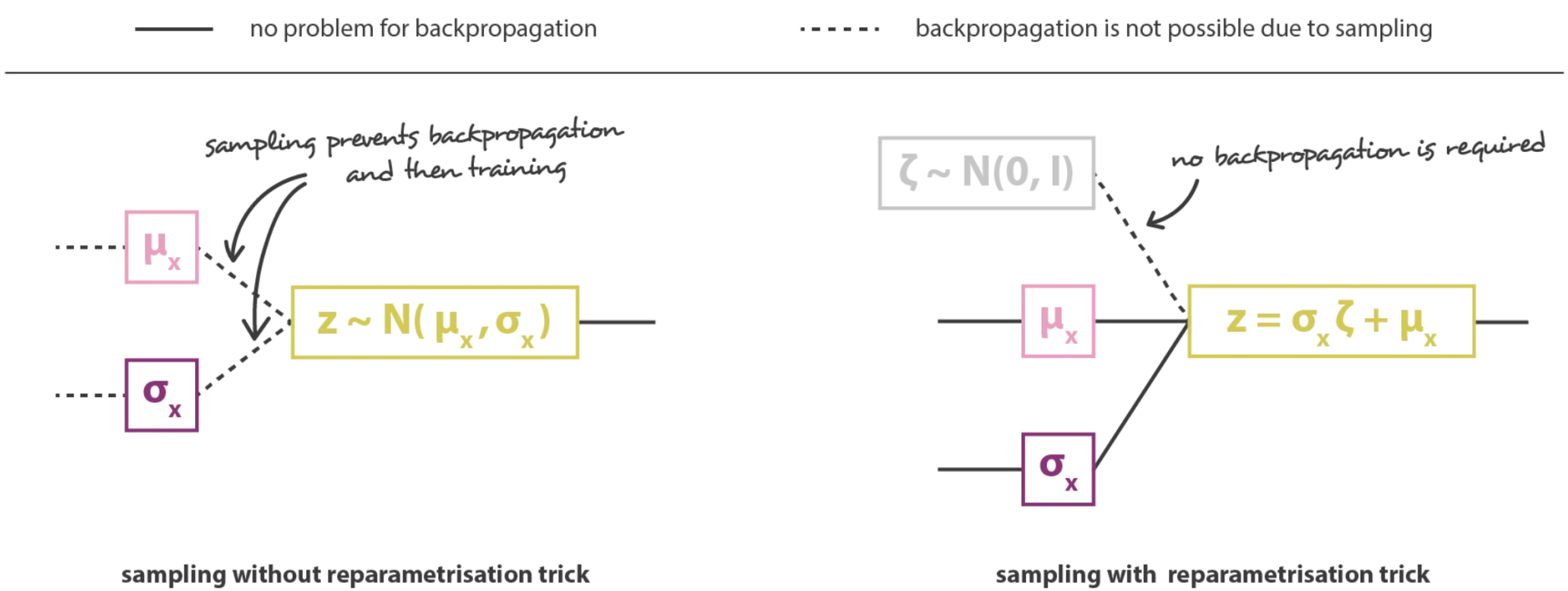
$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}, \boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

The reparameterization trick



Source: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

The reparameterization trick



Source: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

The reparameterization trick

Let z be a continuous random variable, and z ∼ qϕ(z|x) be some conditional distribution. It is then often possible to express the random variable z as a deterministic variable z = gϕ(ε, x), where ε is an auxiliary variable with independent marginal p(ε), and gϕ(.) is some vector-valued function parameterized by ϕ.

For example the gaussion case: $z \sim p(z|x) = N(\mu, \sigma^2)$

Valid reparameterization: $z = \mu + \sigma * \epsilon$

Epsilon is an auxillary noise variabele: $\epsilon \sim N(0,1)$

Approaches for g(.) and ε

1. Tractable inverse CDF: In this case, let $\varepsilon \sim U(0, I)$, and let $g\phi(\varepsilon, x)$ be the inverse CDF of $q\phi(z|x)$. Examples: Exponential, Cauchy, Logistic, Rayleigh, Pareto, Weibull, Reciprocal, Gompertz, Gumbel and Erlang distributions.

2. Analogous to the Gaussian example, for any "location-scale" family of distributions we can choose the standard distribution (with location = 0, scale = 1) as the auxiliary variable ε, and let $g(.) = \text{location} + \text{scale} \cdot \varepsilon$. Examples: Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular and Gaussian distributions.

3. Composition: It is often possible to express random variables as different transformations of auxiliary variables. Examples: Log-Normal (exponentiation of normally distributed variable), Gamma (a sum over exponentially distributed variables), Dirichlet (weighted sum of Gamma variates), Beta, Chi-Squared, and F distributions.

Example: Variational Auto-Encoder

1. The authors used an example with a neural network as a probabilistic encoder, known as $q\phi(z|x)$, approximating the generative model's posterior using jointly optimized parameters ϕ and θ through the AEVB algorithm.
2. They assumed a prior distribution for latent variables, a parameter-less centered isotropic multivariate Gaussian: $p\theta(z) = N(z; 0, I)$.
3. The conditional distribution $p\theta(x|z)$ was modeled as a multivariate Gaussian for real-valued data or a Bernoulli distribution for binary data. Distribution parameters were computed from z using a simple multi-layer perceptron (MLP).
4. Recognizing the intractable true posterior $p\theta(z|x)$, the authors approximated it as a Gaussian distribution with an approximately diagonal covariance structure.
5. This led to the design of a variational approximate posterior, a multivariate Gaussian with a diagonal covariance structure, represented as, $\log q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I})$ (9)

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 $\mathbf{x}^{(i)}$ and the variational parameters $\boldsymbol{\phi}$ (see appendix C).

1. We sampled from the posterior as z(i,l) ~ qϕ(z|x(i)) using z(i,l) = gϕ(x(i), ε(l)) = μ(i) + σ(i) ⊙ ε(l), where ε(l) ~ N(0, I). Here, ⊙ signifies element-wise multiplication.
2. In this model, both pθ(z) (the prior) and qϕ(z|x) are Gaussian. In this scenario, the estimator introduced in equation (We sampled from the posterior as z(i,l) ~ qϕ(z|x(i)) using z(i,l) = gϕ(x(i), ε(l)) = μ(i) + σ(i) ⊙ ε(l), where ε(l) ~ N(0, I). Here, ⊙ signifies element-wise multiplication.
3. In this model, both pθ(z) (the prior) and qϕ(z|x) are Gaussian. In this scenario, the estimator introduced in equation (7) can be employed, where the KL divergence can be computed and differentiated directly without requiring estimation (details in appendix B).
4. The resulting estimator for this model and data point x(i) is:) can be employed, where the KL divergence can be computed and differentiated directly without requiring estimation (details in appendix B).
5. The resulting estimator for this model and data point x(i) is:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \tag{10}$$

RELATED WORK

1. The wake-sleep algorithm as we know, is also the other online learning method that can be applied to the same general class of continuous latent variable models.
2. Similar to our approach, the wake-sleep algorithm utilizes a recognition model to approximate the true posterior.
3. One limitation of the wake-sleep algorithm is that it requires the simultaneous optimization of two objective functions, which do not collectively represent the optimization of a marginal likelihood bound.
4. On the positive side, wake-sleep can also be used with models that have discrete latent variables.
5. Wake-sleep shares the same computational complexity per datapoint as AEVB.

Differences betweeen **Wake-sleep** & VAE(Variational Auto-Encoder)

Objective Function Optimization:

Wake-sleep: Requires concurrent optimization of two objective functions that do not directly correspond to the marginal likelihood.

VAE: Directly optimizes a lower bound on the marginal likelihood.

Applicability to Latent Variables:

Wake-sleep: Applicable to models with both continuous and discrete latent variables.

VAE: Primarily designed for models with continuous latent variables.

Recognition Model:

Wake-sleep: Employs a recognition model to approximate the true posterior, similar to VAE.

VAE: Uses a recognition model (encoder) to approximate the posterior.

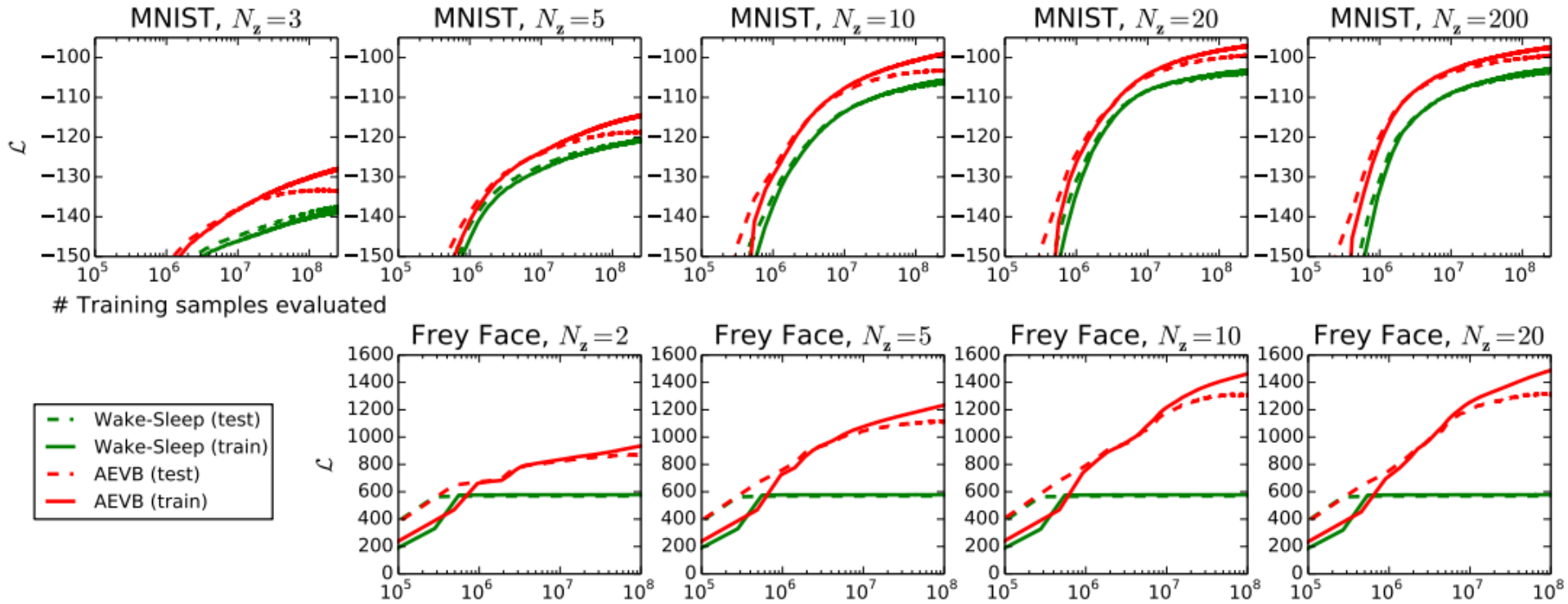
Complexity:

Wake-sleep: Has the same computational complexity as AEVB per data point.

VAE: Computational complexity varies based on architecture and dataset size but also depends on data point computations.

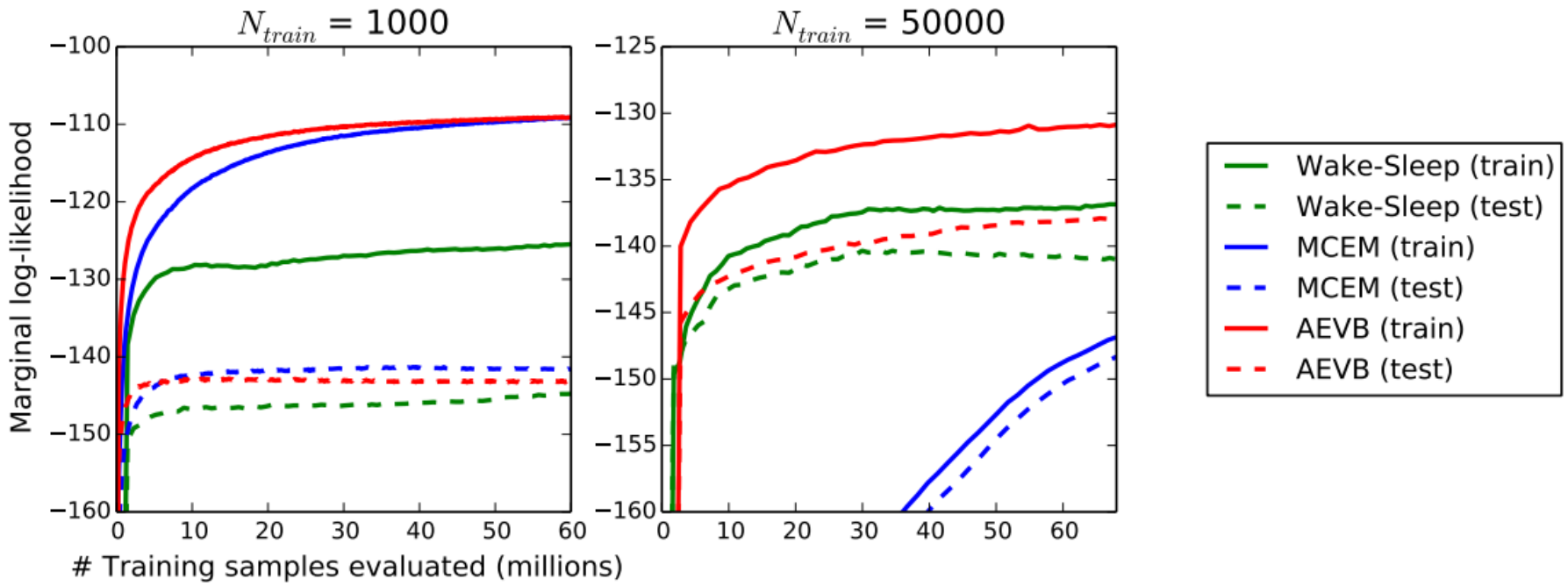
EXPERIMENTS

1. Author compared various learning algorithms for generative models trained on **MNIST** and **Frey Face datasets.**
2. The comparison focused on assessing the performance using the **variational lower bound** and **estimated marginal likelihood criteria.**
3. The generative model utilized an encoder and variational approximation (decoder) with an equal number of hidden units.
4. For the continuous Frey Face dataset, the author compared the use of a Gaussian decoder, which had means constrained to the interval (0, 1) through a sigmoidal activation function.



RESULTS ON MNIST DATASET

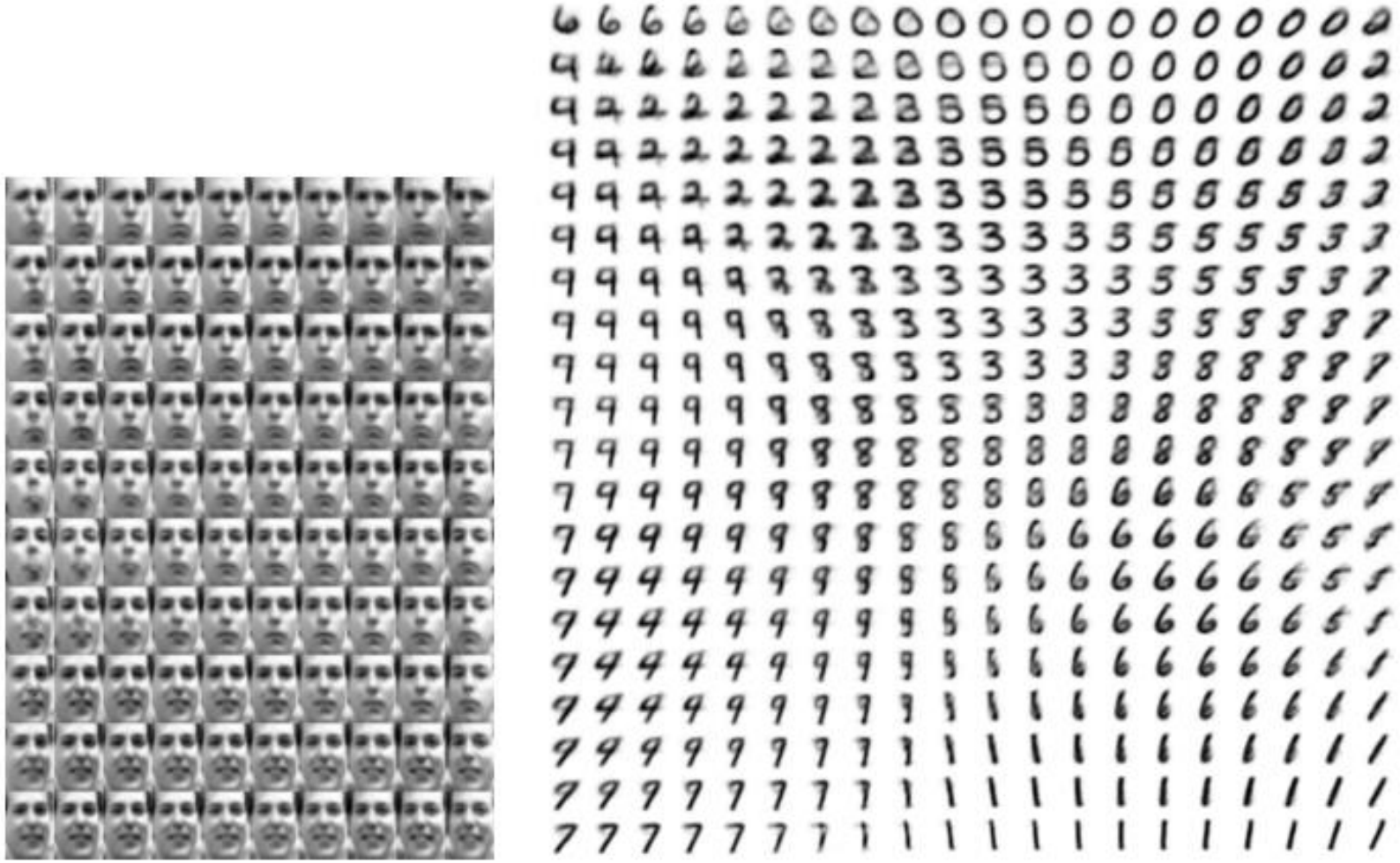
- AEVB method vs. wake-sleep algorithm comparison for lower bound optimization.
- Tested across different latent space dimensions (Nz).
- AEVB achieved faster convergence and consistently better solutions.
- Vertical axis: average variational lower bound per datapoint.
- Minimal estimator variance (< 1), not included.
- Horizontal axis: number of evaluated training points.
- Computation: ~20 minutes per million training samples on a quad-core Xeon CPU.



COMARISION WITH **Monte Carlo EM Algorithm**

- Comparison of AEVB, wake-sleep algorithm, and Monte Carlo EM for estimated marginal likelihood.
- Evaluation across different numbers of training points.
- Monte Carlo EM is not suitable for full MNIST dataset, unlike AEVB and wake-sleep.
- Method efficiency and performance were assessed.
- Results highlighted the scalability and effectiveness of AEVB and wake-sleep methods compared to Monte Carlo EM.
- Highlight the challenges faced by Monte Carlo EM with large datasets like MNIST.

Visualisations of learned data manifold for generative models with two dimensional latentspace, learned with AEVB.



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Random samples from learned generative models of MNIST for different dimensionalities of latent space.

(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

CONCLUSION

1. The authors introduced a new technique called Stochastic Gradient VB (SGVB) to help estimate the variational lower bound when dealing with continuous latent variables.

2. They also developed a method called Auto-Encoding VB (AEVB) specifically for datasets where the samples are independent and identically distributed (I.I.D.), and each sample has continuous latent variables. AEVB uses SGVB to learn an approximate inference model.

3. The practical success of SGVB and AEVB was demonstrated through experiments.

Future Prospects

Since the SGVB estimator and AEVB method can work for many different problems involving continuous hidden variables, there are several exciting future possibilities:

1. Creating complex generative systems with deep neural networks (like convolutional networks) for the encoders and decoders. These networks would be trained together with AEVB.
2. Exploring time-related models, like dynamic Bayesian networks.
3. Applying SGVB to handle the overarching parameters.
4. Developing models that can learn from data with hidden variables, which can be handy for handling complex variations in data patterns.

"Materials that aided in comprehending the paper."

Understanding Variational Autoencoders (VAEs)

Variational Auto encoders by Ahlad kumar

VAE: Variational Auto-Encoder by Naoki

THANK YOU !