Stochastic Gradient VB

Intractable posterior distributions?
Gradients to the Rescue!

An efficient variational inference algorithm for the intractable case.

Durk Kingma
1st year PhD Candidate


UNIVERSITEIT VAN AMSTERDAM

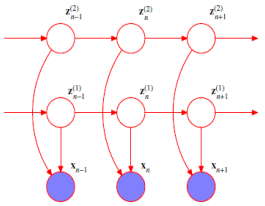
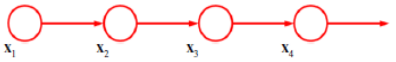
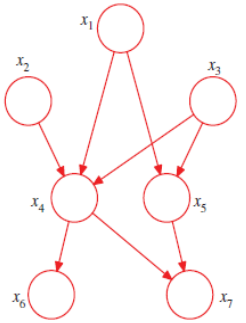
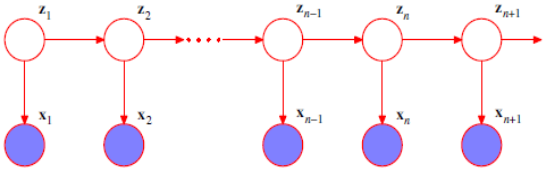


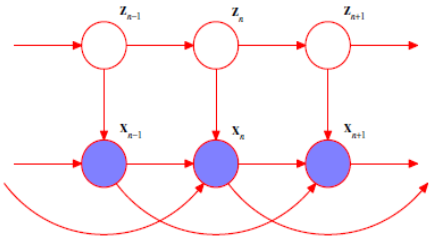
Table of contents

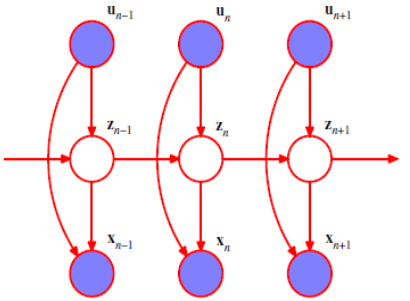




- Introduction
  - Bayesian posterior inference
- Method
  - Our estimator of the bound
  - Parameterization with a noise variable
  - Application: the variational auto-encoder
- Experiments
- Conclusion



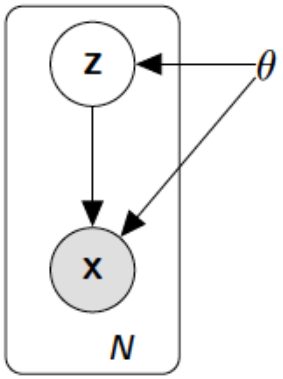




General setup

- General setup:
  - **x** : observed variables
  - **z** : unobserved variables (continuous)
  - **θ** : model parameters
  - A specified joint PDF: $p_{\theta}(\mathbf{x},\mathbf{z})$
  - posterior $p_{\theta}(\mathbf{z}|\mathbf{x})$ is *intractable*
- **We want to:**
  - Infer an (approximate) posterior distribution over the latent variables **z**
  - Learn the parameters **θ** (obtain a MAP estimate)
    OR infer an approximate posterior distribution over the parameters

e.g:



- What if the posterior $p_\theta(z \mid x)$ is intractable?
  - We cannot use standard EM
  - Sampling-based EM (MCEM) is often still possible, but requires sampling from the latent variables' posterior, which is slow
  - Mean-field VB is often also impossible (it requires closed-form solutions to certain expectations of the joint PDF)

- **Proposed solution: Stochastic VB**

The Variational Bound

- We introduce the variational approximation: $q_{\boldsymbol{\phi}}(\mathbf{z})$
  - Distribution can be almost anything (we use Gaussian)
  - Will approximate the true (but intractable) posterior

Marginal likelihood can be written as:

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

$$\text{where} \quad \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})\right) d\mathbf{z}$$

$$\left(\text{the variational lower bound of the marg. likelihood}\right)$$

This bound is exactly we want to optimize!
(w.r.t. **φ** and **θ)**

“Naive” Monte Carlo estimator of the bound

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}) \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}) \right] d\mathbf{z}$$

$$\simeq \frac{1}{L} \sum_{l=1}^{L} \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}^{(l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)}) \right)$$

$$\text{where} \quad \mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}) \quad (\text{samples})$$

Problem: not differentiable w.r.t. **φ! (**Cannot differentiate through sampling process).

Recently proposed solutions (2013)

- by Tim Salimans: (**only applies to Exponential Family q**)
- Michael Jordan / David Blei (**very high variance**)

Key “reparameterization trick”: Alternative way of sampling from q(z)

Two-step sampling process:

1. First sample 'ε' from p(ε) *(independent of φ!)*
2. $z = g(\varphi, \varepsilon)$ (some differentiable function of φ and ε)

   Such that $z \sim q_\varphi(z)$ (the correct distribution)

Examples:

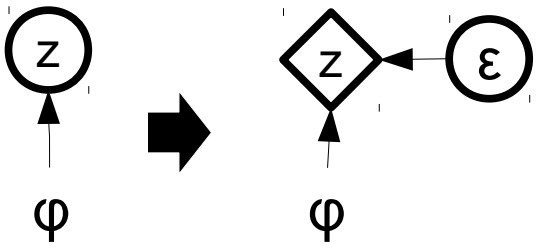
- if $q(z) \sim N(\mu, \sigma)$

  $\varepsilon \sim N(0,1)$

  $z = \mu + \sigma * \varepsilon$
- Inverse CDF:

  $\varepsilon \sim U(0,1)$

  $z = F^{-1}(\varphi, \varepsilon)$



SGVB estimator

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}) \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}) \right] d\mathbf{z}$$

$$\simeq \frac{1}{L} \sum_{l=1}^{L} \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}^{(l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)}) \right)$$

$$\text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \quad \text{(samples from noise variable)}$$

$$\mathbf{z}^{(l)} = g(\boldsymbol{\epsilon}^{(l)}, \boldsymbol{\phi})$$

$$\text{(such that} \quad \mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}))$$

Really simple and differentiable w.r.t. **φ** and **θ!**

Basic SGVB Algorithm (L=1)

repeat

$$\boldsymbol{\epsilon} \leftarrow \text{sample from } p(\boldsymbol{\epsilon})$$

$$g_{\boldsymbol{\theta}} \leftarrow \nabla_{\boldsymbol{\theta}} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, g(\boldsymbol{\epsilon}, \boldsymbol{\phi}))$$

$$g_{\boldsymbol{\phi}} \leftarrow \nabla_{\boldsymbol{\phi}} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, g(\boldsymbol{\epsilon}, \boldsymbol{\phi}))$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot g_{\boldsymbol{\theta}}$$

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \cdot g_{\boldsymbol{\phi}}$$

until convergence

where $\widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{z}^{(l)}) = \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}^{(l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})$

Example: isotropic Gaussian *q*

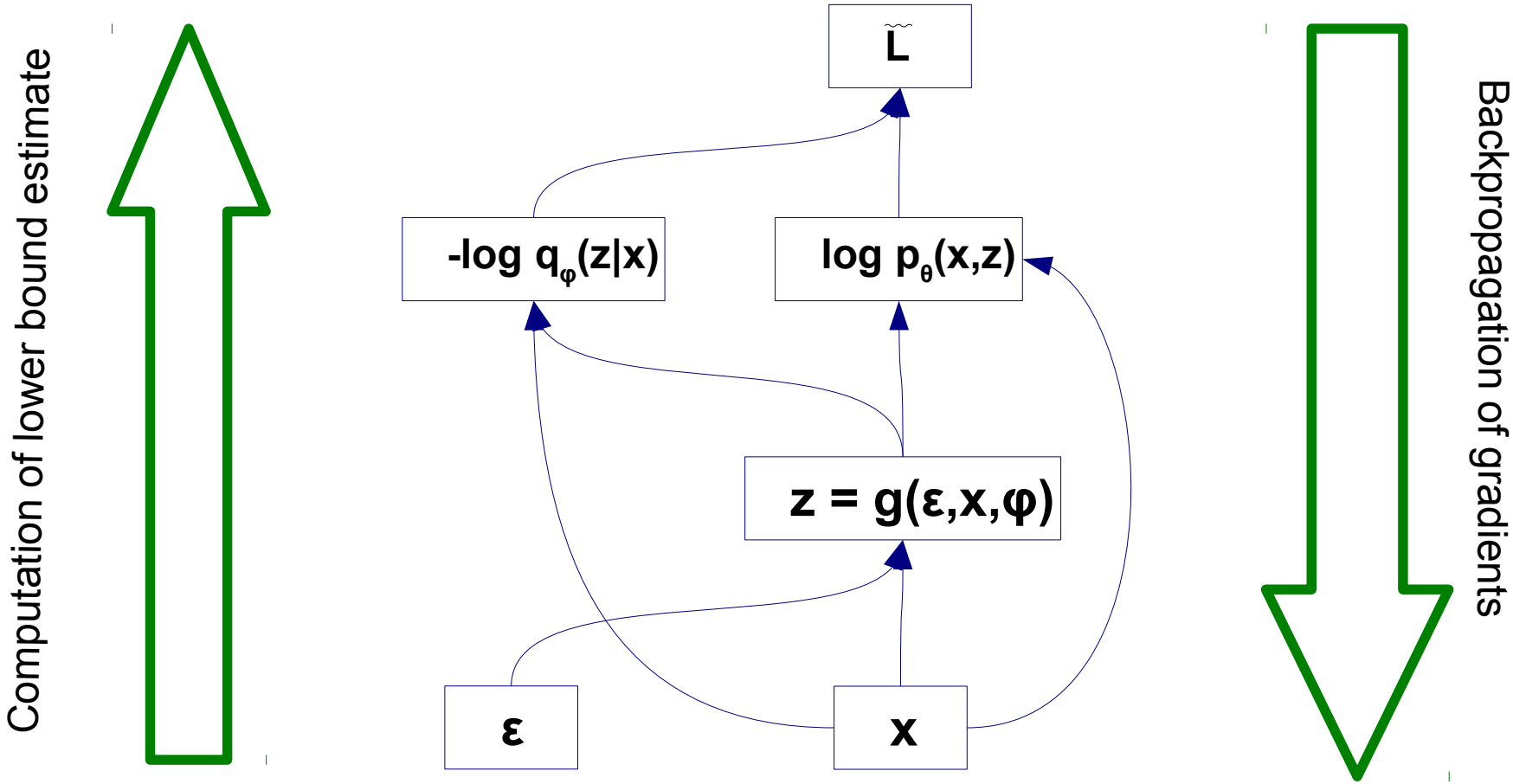
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}) \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}) \right] d\mathbf{z}$$

$$\simeq \frac{1}{L} \sum_{l=1}^{L} \left(\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}^{(l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)}) \right)$$

where $\quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \quad$ (samples from noise variable)

$$\mathbf{z}^{(l)} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}^{(l)}$$

Backprop: computing gradients efficiently

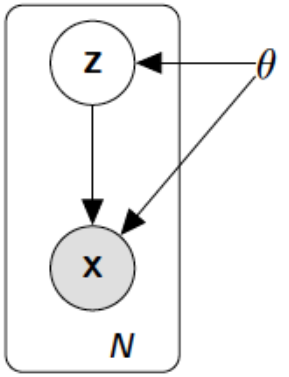


python + theano

AEVB (Auto-Encoding VB): efficient on-line version of SGVB

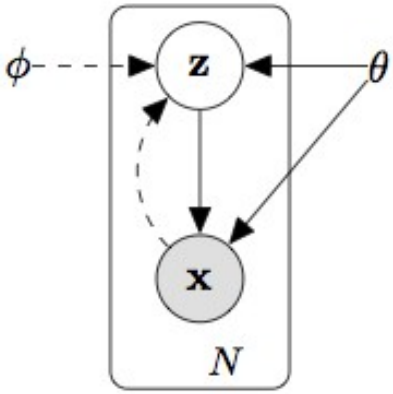
- Special case of SGVB:

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}) p_{\boldsymbol{\theta}}(\mathbf{z}^{(i)})$$



Large i.i.d. dataset (large N)

Avoid local parameters!



- AEVB solution: $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$
  - Use conditional:
  - Doubly stochastic optimization

Basic AEVB algorithm (L=1)

repeat

$$\mathbf{x} \leftarrow \text{random minibatch from dataset}$$

$$\boldsymbol{\epsilon} \leftarrow \text{sample from } p(\boldsymbol{\epsilon})$$

$$g_{\boldsymbol{\theta}} \leftarrow \nabla_{\boldsymbol{\theta}} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, g(\boldsymbol{\epsilon}, \boldsymbol{\phi}))$$

$$g_{\boldsymbol{\phi}} \leftarrow \nabla_{\boldsymbol{\phi}} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, g(\boldsymbol{\epsilon}, \boldsymbol{\phi}))$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot g_{\boldsymbol{\theta}}$$

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \cdot g_{\boldsymbol{\phi}}$$

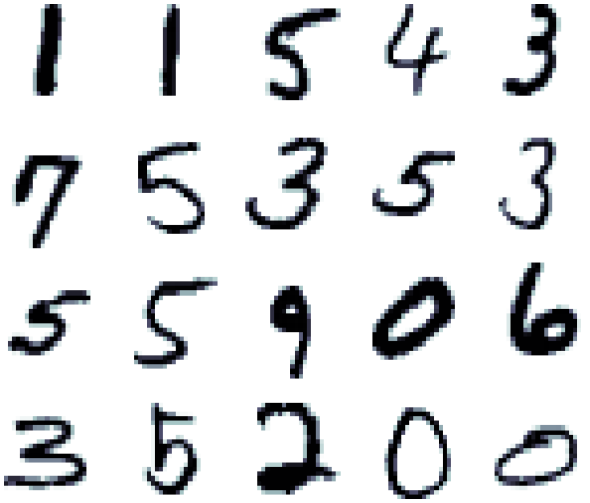
until convergence

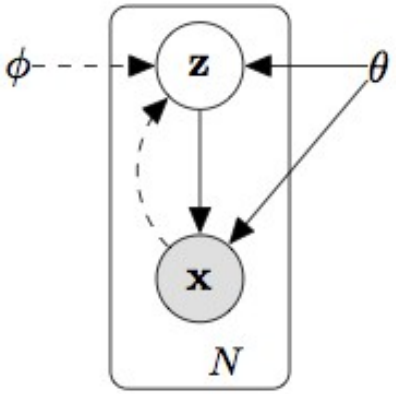
Experiments

- Model:
  - p(z): Gaussian
  - p(x|z): Bernoulli or Gaussian whose parameters are determined by MLP (neural network).

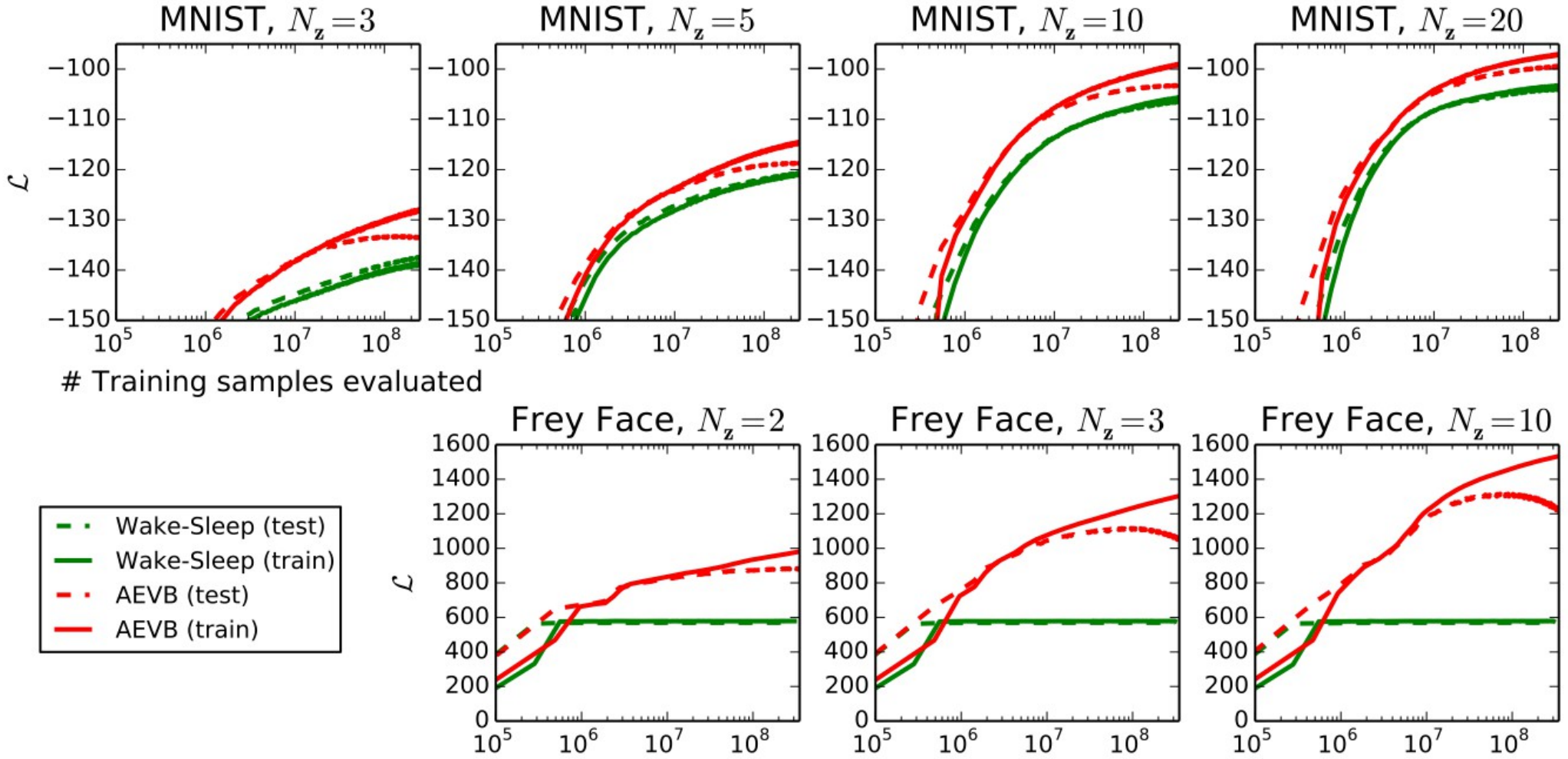
    “decoder”
- Posterior approximation:
  - q(z|x): Gaussian whose parameters are also determined by MLP

    “encoder”

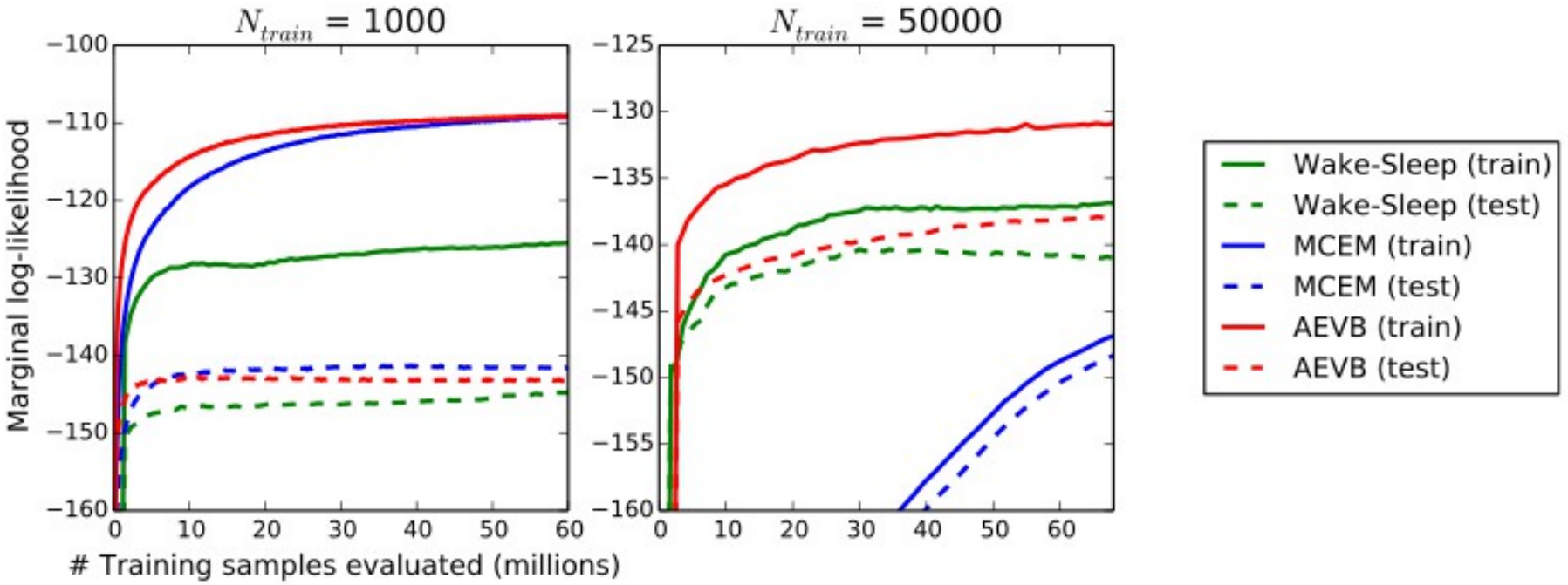




Results:
Marginal likelihood lower bound

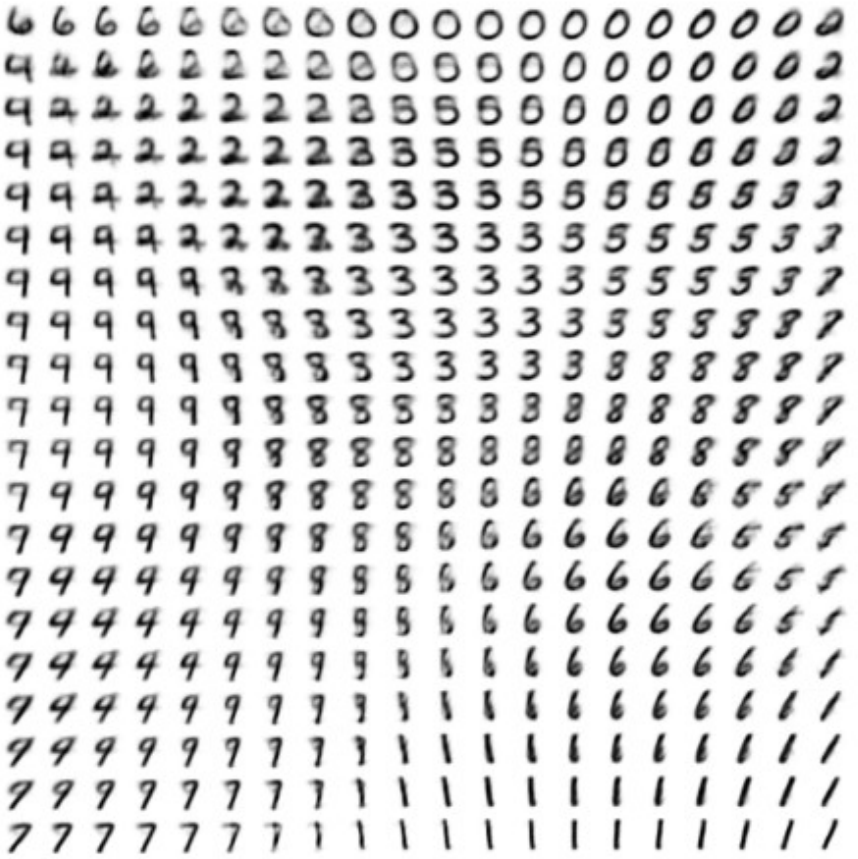


Results: Marginal log-likelihood



learned 2D manifolds

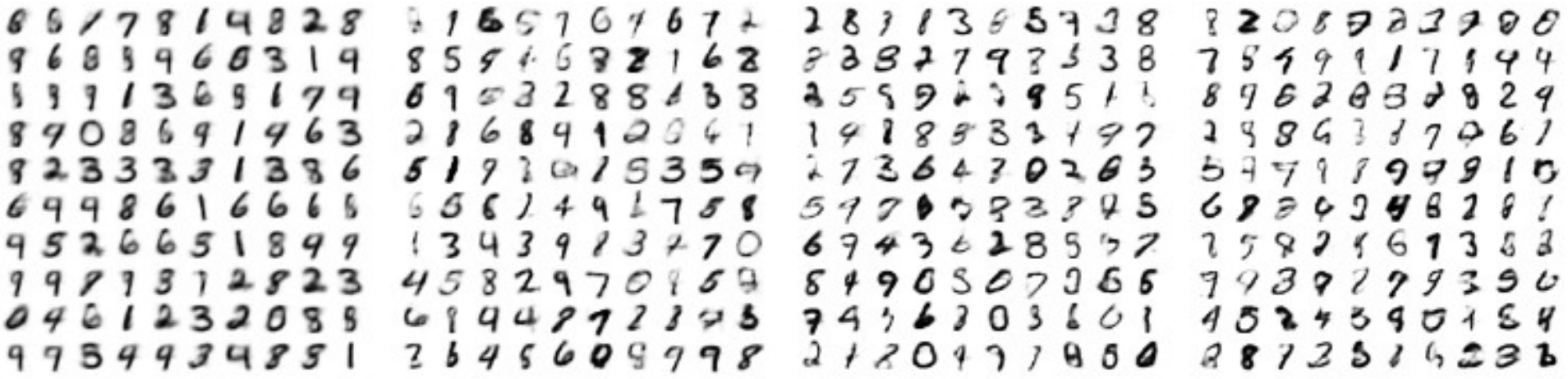
(a) Learned Frey Face manifold



(b) Learned MNIST manifold

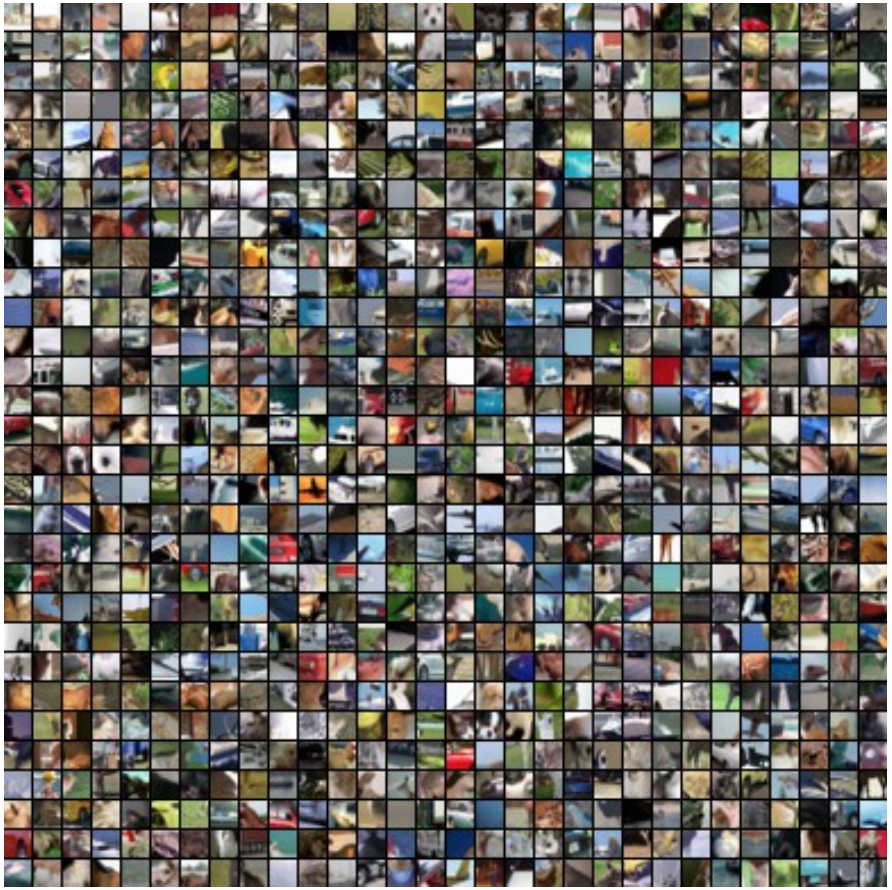
learned 3D manifold

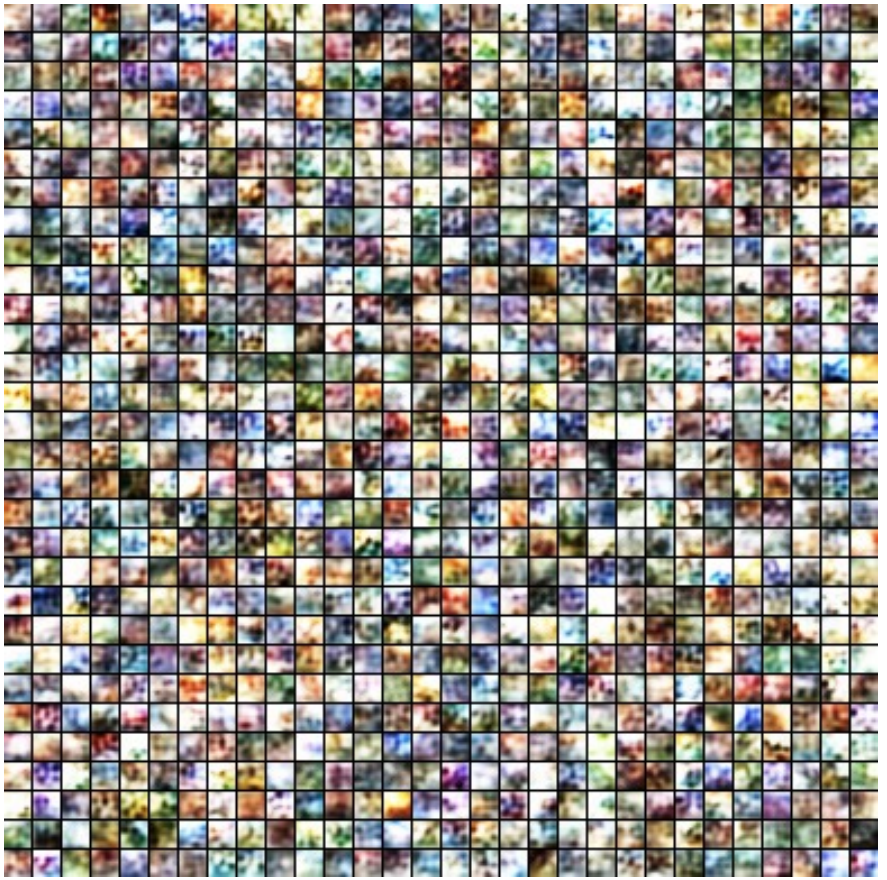
Samples from MNIST



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

12x12 image patches form CIFAR-10





Conclusion & future work

SGVB: general algorithm for inference and learning with continuous latent variables

AEVB: version for big datasets

- Future work:
  - Experiments with general SGVB algo
  - Other models (deep learning) (AEVB), time series models
  - Example future directions Applications:
    - Alternative to other unsupervised learning algorithms (RBM's)
    - Representation learning / semi-supervised learning
    - Bayesian inference for images: denoising, completion (infilling), etc.
    - More complex models (time series, convolutional, deep models)

Thanks!