Auto-Encoding Variational Bayes

Diederik P. Kingma and Max Welling

Ashish Bora

Oct 2016

Outline

Bayesian Learning

- Observations x, Parameters/Unobserved variables z
  Example

x =  , z = cat

Image from Google Images

Bayesian Learning

- Observations x, Parameters/Unobserved variables z
- Likelihood $= p(x \mid z)$

Image from Google Images

Bayesian Learning

- Observations x, Parameters/Unobserved variables z
- Likelihood $= p(x \mid z)$
- Prior $= p(z)$

p(cat)

Bayesian Learning

- Observations x, Parameters/Unobserved variables z
- Likelihood $= p(x \mid z)$
- Prior $= p(z)$
- Joint $= p(x, z) = p(x \mid z)p(z)$

p( , cat)

Image from Google Images

Bayesian Learning

- Observations x, Parameters/Unobserved variables z
- Likelihood $= p(x \mid z)$
- Prior $= p(z)$
- Joint $= p(x, z) = p(x \mid z)p(z)$
- Marginal Likelihood $= p(x) = \int_z p(x \mid z)p(z)dz$

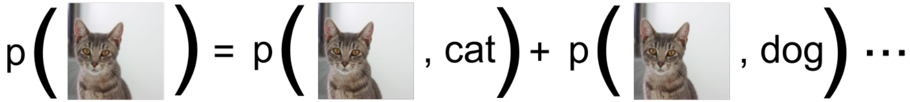


Image from Google Images

- Observations x, Parameters/Unobserved variables z
- Likelihood $= p(x \mid z)$
- Prior $= p(z)$
- Joint $= p(x, z) = p(x \mid z)p(z)$
- Marginal Likelihood $= p(x) = \int_z p(x \mid z)p(z)dz$
- Posterior $= p(z \mid x)$

Image from Google Images

Bayesian Learning

- Observations x, Parameters/Unobserved variables z
- Likelihood $= p(x \mid z)$
- Prior $= p(z)$
- Joint $= p(x, z) = p(x \mid z)p(z)$
- Marginal Likelihood $= p(x) = \int_z p(x \mid z)p(z)dz$
- Posterior $= p(z \mid x)$
- Non-Bayesian Learning : Maximize likelihood wrt parameters z

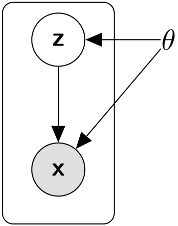
$$p(x \mid z)$$

- Bayesian Learning : Maximize marginal likelihood with a prior $p(z)$ on parameters z

$$p(x)$$

Latent Variable Models

- Parameters: θ, Latent variables: z, Observations x
- Two step process:

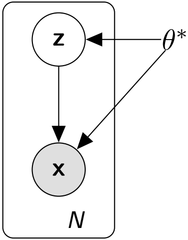


- $z \sim p_{\boldsymbol{\theta}}(\mathbf{z})$
- Given z, $x \sim p_{\boldsymbol{\theta}}(\mathbf{x} \mid z)$
- We only observe $x \sim p_{\boldsymbol{\theta}}(\mathbf{x})$
- $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int_z p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}) d\mathbf{z}$.

Tasks

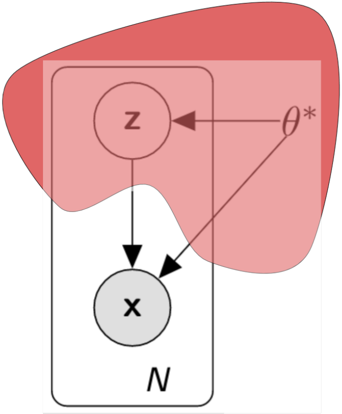
- ML/MAP inference on θ.
  mimic data generation
- Posterior inference on z given x.
  coding/data representation.
- Marginal inference on x
  denoising, inpainting, super-resolution.

Want efficient algorithms for all, with minimal assumptions.

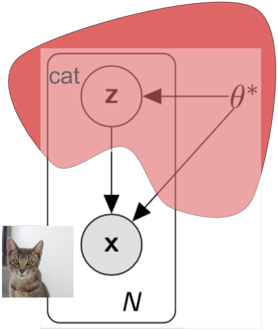


Why is this hard?

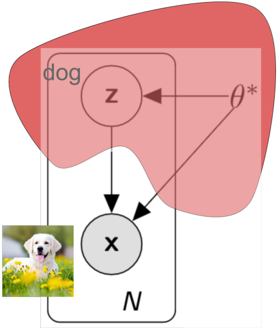
- Lots of stuff hidden from us, we only see x.
- Data generation process can be complicated



Ideas?



,



...

Given only images, we want

- θ close to θ^*
- An approximation to $p_{\theta^*}(z \mid x)$
- A good model of $p_{\theta^*}(x)$

Images from Google Images

Some approaches

Idea 1

- Integrate out z. Maximize marginal likelihood

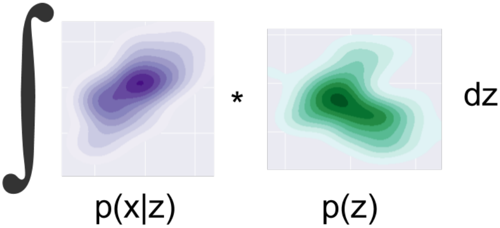
$$p_\theta(\mathbf{x}) = \int_z p_\theta(x \mid z) p_\theta(z) dz$$

.

- Integrate out z. Maximize marginal likelihood

$$p_\theta(\mathbf{x}) = \int_z p_\theta(x \mid z) p_\theta(z) dz$$

.
Problem : nasty integral



Some approaches

Idea 2

- Alternating optimization between z and θ : Expectation Maximization.

  Example : kMeans

  - θ contains cluster centres.
  - for every x_i, z_i is the cluster id.

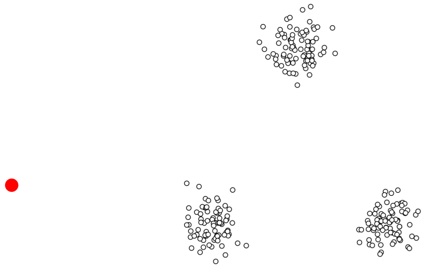


Image from https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

- Alternating optimization between z and θ : Expectation Maximization.

  Example : kMeans
  - θ contains cluster centres.
  - for every x_i, z_i is the cluster id.

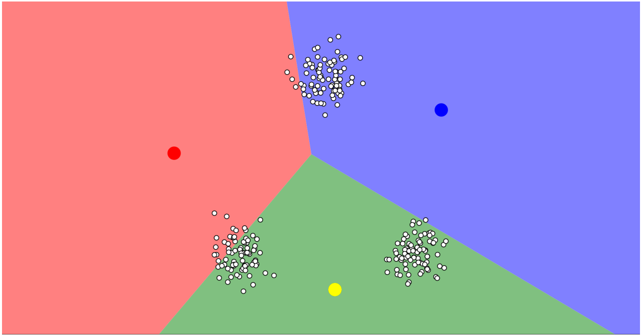


Image from https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Some approaches
Idea 2

- Alternating optimization between z and θ : Expectation Maximization.

  Example : kMeans
  - θ contains cluster centres.
  - for every x_i, z_i is the cluster id.

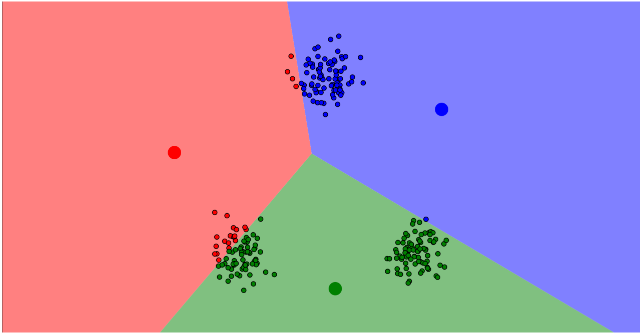


Image from https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

- Alternating optimization between z and θ : Expectation Maximization.

  Example : kMeans
  - θ contains cluster centres.
  - for every x_i, z_i is the cluster id.

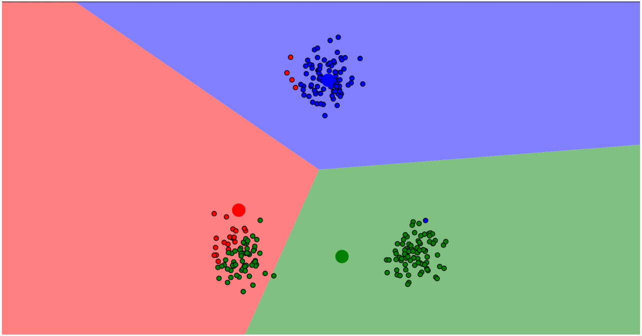


Image from https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

- Alternating optimization between z and θ : Expectation Maximization.

  Example : kMeans
  - θ contains cluster centres.
  - for every x_i, z_i is the cluster id.

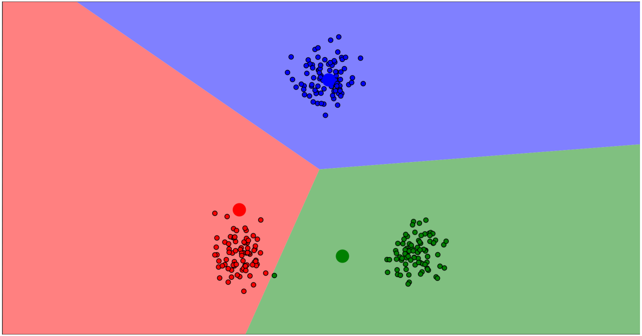


Image from https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Some approaches
Idea 2

- Alternating optimization between z and θ : Expectation Maximization.

  Example : kMeans
  - θ contains cluster centres.
  - for every x_i, z_i is the cluster id.

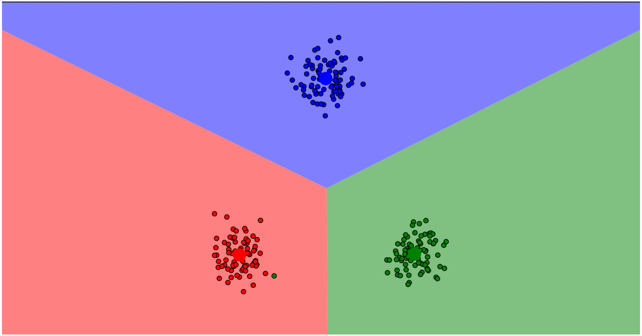


Image from https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Some approaches

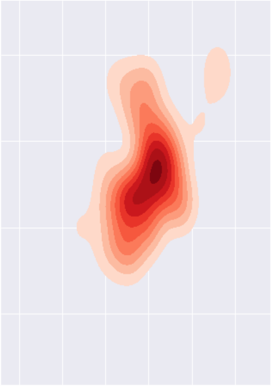
Idea 2

- Alternating optimization between z and θ : Expectation Maximization.

  Problem : Posterior $p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$ may not be tractable.

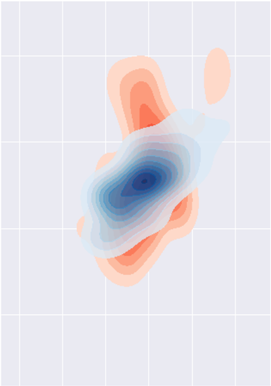
Enter Variational Inference!

- Problem: We want to estimate some distribution $p_{\boldsymbol{\theta}}(\cdot)$, but direct estimation is hard.



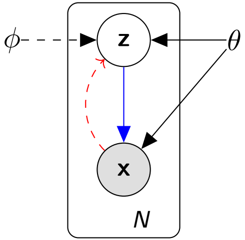
Enter Variational Inference!

- Problem: We want to estimate some distribution $p_{\boldsymbol{\theta}}(\cdot)$, but direct estimation is hard.
- Solution
  - Approximate $p_{\boldsymbol{\theta}}(\cdot)$ with a simpler distribution $q_{\phi}(\cdot)$.
  - Find parameters ϕ such that the approximation is "close".



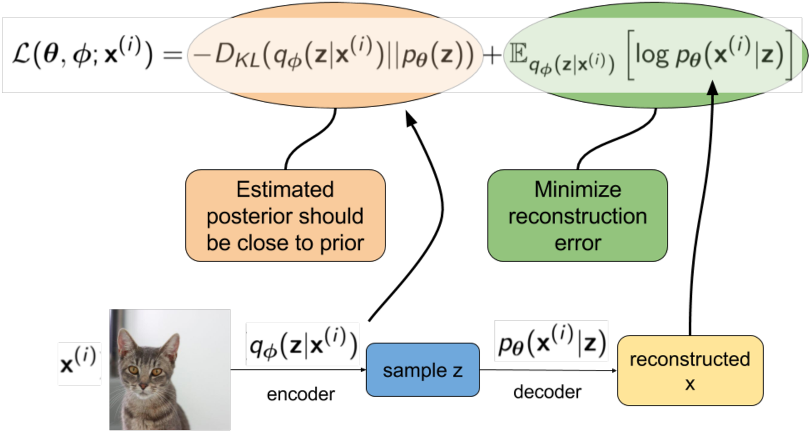
Variational Inference – our setting

- Since the posterior $p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$ is intractable, use variational approximation $q_{\phi}(\mathbf{z} \mid \mathbf{x})$.
- $q_{\phi}(\mathbf{z} \mid \mathbf{x})$: probabilistic encoder
- $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$: probabilistic decoder



Variational Lower Bound

- Using approximation leads to smaller marginal likelihood.
- Lower bound on marginal likehood in terms of the variational approximation:



Where is Deep Learning?

Image taken Google Images (modified)

Parametrizing distributions

Ideas?

Parametrizing distributions

Ideas?

- Since we did not assume tractable posteriors, can use any artibrary functions for generative and variational part.
- Only requirement - we should be able to optimize wrt θ and ϕ.
- For gradient based algorithms, we want a paramteric family which is differentiable wrt inputs and parameters.
- Can use neural networks.

Parametrizing distributions

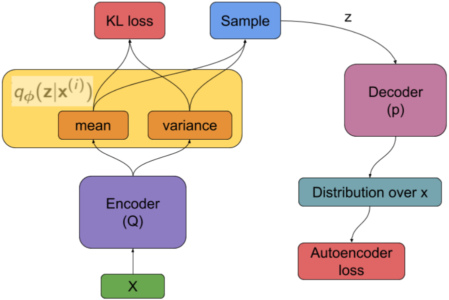
Ideas?

- Since we did not assume tractable posteriors, can use any artibrary functions for generative and variational part.
- Only requirement - we should be able to optimize wrt θ and ϕ.
- For gradient based algorithms, we want a paramteric family which is differentiable wrt inputs and parameters.
- Can use neural networks.

Image from Google Images

Example : Variational Autoencoder

- $p_{\theta}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$
- $p_{\theta}(\mathbf{x}|\mathbf{z})$ be a "simple" distribution whose distribution parameters are computed from $\mathbf{z}$ with a neural network.
- Assume $q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I})$, where μ, and σ are predicted using a neural network



Gradient based optimization : Naïve method

- Lower bound is

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) &= -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\right]
\end{aligned}
$$

- Monte Carlo estimator:

$$
\begin{aligned}
\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\right] &= \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z}) \nabla_{q_\phi(\mathbf{z})} \log q_\phi(\mathbf{z})\right] \\
&\simeq \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)}) \nabla_{q_\phi(\mathbf{z}^{(l)})} \log q_\phi(\mathbf{z}^{(l)})
\end{aligned}
$$

$$
\text{where} \quad \mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})
$$

- This has very high variance.

Gradient based optimization : Naïve method

- Lower bound is

$$\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$$
$$= \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\right]$$

- Monte Carlo estimator:

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\right] = \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\nabla_{q_\phi(\mathbf{z})} \log q_\phi(\mathbf{z})\right]$$
$$\simeq \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})\nabla_{q_\phi(\mathbf{z}^{(l)})} \log q_\phi(\mathbf{z}^{(l)})$$
$$\text{where} \quad \mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$$

- This has very high variance. Q : Why?

Gradient based optimization : Naïve method

- Lower bound is

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) &= -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\right]
\end{aligned}
$$

- Monte Carlo estimator:

$$
\begin{aligned}
\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\right] &= \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\nabla_{q_\phi(\mathbf{z})} \log q_\phi(\mathbf{z})\right] \\
&\simeq \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})\nabla_{q_\phi(\mathbf{z}^{(l)})} \log q_\phi(\mathbf{z}^{(l)})
\end{aligned}
$$

$$
\text{where} \quad \mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})
$$

- This has very high variance. Q : Why?
  A : Gradient of log of probability. Probability very close to zero means very large values.

The reparametrization trick

- Problem in naïve method : learnable parameters responsible for producing probabilities.
- Observation: We don't need those probabilities, just an expectation taken using them.
- Solution
  - Instead of producing probability for each z, produce z directly
  - Make sure the distribution of z is the same.
  - For randomness in z generation, use a deterministic function with noise as input. i.e.

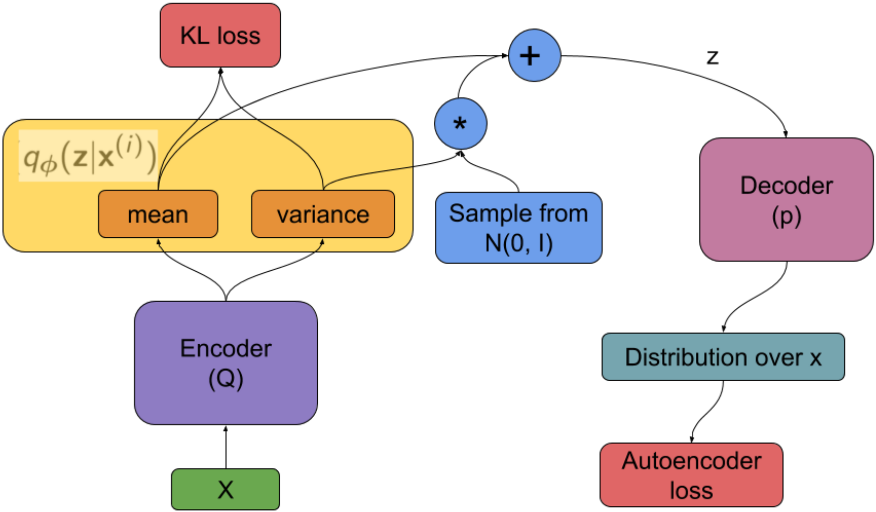
$$g_\phi(z, \epsilon) = \tilde{z} \sim q_\phi(z|x), \ \epsilon \sim p(\epsilon)$$

- Now we can write expectations in terms of $p(\epsilon)$.

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[f(\mathbf{z})\right] = \mathbb{E}_{p(\epsilon)}\left[f(g_\phi(\boldsymbol{\epsilon}, \mathbf{x}^{(i)}))\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)}))$$

where $\quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$

Reparametrization for VAE



Putting it together

- Loss function:

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

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

- Algorithm

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
    $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
    $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
    $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$
    $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

- Experiments on MNIST and FRAY face dataset
- Metric
  - Likelihood lower bound, same as optimization objective.

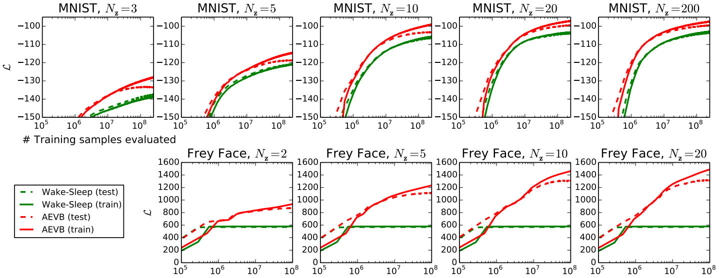


Image from https://arxiv.org/abs/1312.6114

- Experiments on MNIST and FRAY face dataset
- Metric
  - Estimated Marginal Likelihood

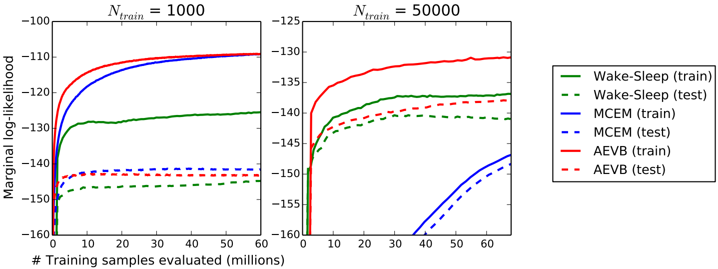
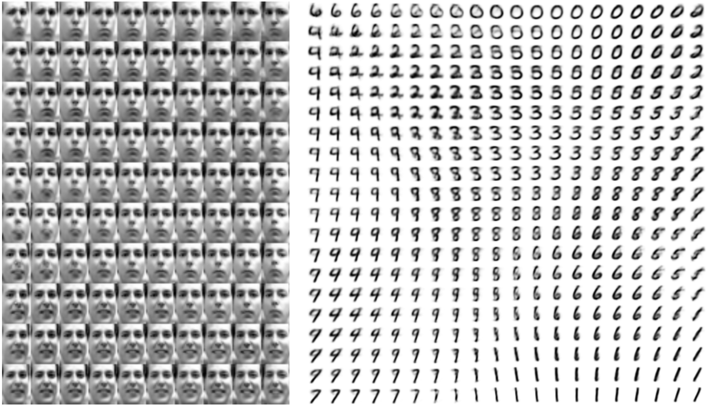


Image from https://arxiv.org/abs/1312.6114

Variational Autoencoder

Generated Samples



Images from https://arxiv.org/abs/1312.6114

Variational Autoencoder

Generated Samples

Image from http://torch.ch/blog/2015/11/13/gan.html

MNIST VAE Demo

https://transcranial.github.io/keras-js

Summary

- Problem : Latent variable model with intractable posterior, large dataset
- Solution
  - Variational approximation for posterior
  - Optimize variational lower bound
  - Reparametrize recognition model to reduce variance
  - Can plug in any function approximator (e.g. neural network)
- Example application
  - Variational Autoencoder

Discussion points

- What are the limitations of VAEs?
- How do they compare to GANs?
- VAE output images more blurred as compared to GANs
- Other questions?

Thanks!