Importance Weighted Autoencoders with Uncertain Neural Network Weights

Daniel Hernández–Lobato
Computer Science Department
Universidad Autónoma de Madrid

http://dhnzl.org, daniel.hernandez@uam.es

Joint work with Thang D. Bui, Yingzhen Li, José Miguel Hernández–Lobato and Rich E. Turner.

Unsupervised Learning and Generative Models

Given some data $\{\mathbf{x}_i\}_{i=1}^N$ we want to estimate $p(\mathbf{x})$.

Unsupervised Learning and Generative Models

Given some data $\{\mathbf{x}_i\}_{i=1}^N$ we want to estimate $p(\mathbf{x})$.

- **Generate additional data.**

Unsupervised Learning and Generative Models

Given some data $\{\mathbf{x}_i\}_{i=1}^N$ we want to estimate $p(\mathbf{x})$.

- **Generate additional data.**
- **Better understand the observed data.**

Unsupervised Learning and Generative Models

Given some data $\{\mathbf{x}_i\}_{i=1}^{N}$ we want to estimate $p(\mathbf{x})$.

- **Generate additional data.**
- **Better understand the observed data.**

"What I cannot create, I do not understand."

—Richard Feynman

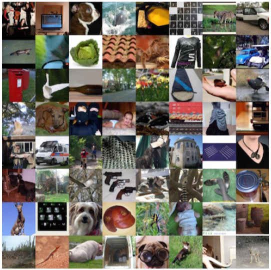
Unsupervised Learning and Generative Models

Given some data $\{\mathbf{x}_i\}_{i=1}^{N}$ we want to estimate $p(\mathbf{x})$.

- **Generate additional data.**
- **Better understand the observed data.**

"What I cannot create, I do not understand."

—Richard Feynman



Real Images

Generated Images

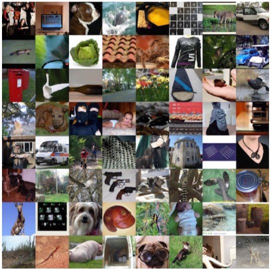
Unsupervised Learning and Generative Models

Given some data $\{\mathbf{x}_i\}_{i=1}^{N}$ we want to estimate $p(\mathbf{x})$.

- **Generate additional data.**
- **Better understand the observed data.**

"What I cannot create, I do not understand."

—Richard Feynman



Real Images

Generated Images

Latent Variable Models

It may be easier to generate first a **latent variable $\mathbf{z}$** and then the data $\mathbf{x}$.

Latent Variable Models

It may be easier to generate first a **latent variable $\mathbf{z}$** and then the data $\mathbf{x}$.

Latent Variable Models

It may be easier to generate first a **latent variable z** and then the data **x**.

Latent Variable Models

It may be easier to generate first a **latent variable z** and then the data **x**.

Latent Variable Models

It may be easier to generate first a **latent variable z** and then the data **x**.

The latent variable **z**:

Latent Variable Models

It may be easier to generate first a **latent variable z** and then the data **x**.

The latent variable **z**:

- **Captures high-level information about x.**

Latent Variable Models

It may be easier to generate first a **latent variable z** and then the data **x**.

The latent variable **z**:

- **Captures high-level information about x.**
- **Compressed representation of x.**

Latent Variable Models

It may be easier to generate first a **latent variable $\mathbf{z}$** and then the data $\mathbf{x}$.

The latent variable $\mathbf{z}$:

- **Captures high-level information about x.**
- **Compressed representation of x.**

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

A Model that can Explain the Observed Data

We consider $p(\mathbf{z})$ is something simple we can sample from. Can we generate one $\mathbf{x}$ similar to each $\{\mathbf{x}_i\}_{i=1}^{N}$ using a parametric $p(\mathbf{x}|\mathbf{z};\theta)$?

A Model that can Explain the Observed Data

We consider $p(\mathbf{z})$ is something simple we can sample from. Can we generate one $\mathbf{x}$ similar to each $\{\mathbf{x}_i\}_{i=1}^N$ using a parametric $p(\mathbf{x}|\mathbf{z};\theta)$?

Yes, if $p(\mathbf{x}|\mathbf{z};\theta)$ has enough flexibility.

A Model that can Explain the Observed Data

We consider $p(\mathbf{z})$ is something simple we can sample from. Can we generate one $\mathbf{x}$ similar to each $\{\mathbf{x}_i\}_{i=1}^N$ using a parametric $p(\mathbf{x}|\mathbf{z};\theta)$?

Yes, if $p(\mathbf{x}|\mathbf{z};\theta)$ has enough flexibility.

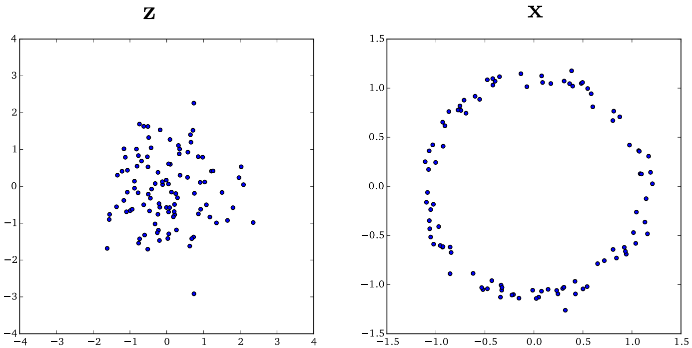
$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ $\quad\quad$ $\mathbf{x} = \mathbf{z}/10 + \mathbf{z}/|\mathbf{z}|$

A Model that can Explain the Observed Data

We consider $p(\mathbf{z})$ is something simple we can sample from. Can we generate one $\mathbf{x}$ similar to each $\{\mathbf{x}_i\}_{i=1}^N$ using a parametric $p(\mathbf{x}|\mathbf{z};\theta)$?

Yes, if $p(\mathbf{x}|\mathbf{z};\theta)$ has enough flexibility.

$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ $\qquad$ $\mathbf{x} = \mathbf{z}/10 + \mathbf{z}/|\mathbf{z}|$



Training the Model

Let $p(\mathbf{x}|\mathbf{z};\theta)$ be a factorizing Gaussian with parameters given by a MLP.

$$p(\mathbf{x}|\mathbf{z};\theta) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d(\mathbf{z};\theta), \sigma_d^2(\mathbf{z};\theta))$$

Training the Model

Let $p(\mathbf{x}|\mathbf{z};\theta)$ be a factorizing Gaussian with parameters given by a MLP.

$$p(\mathbf{x}|\mathbf{z};\theta) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d(\mathbf{z};\theta), \sigma_d^2(\mathbf{z};\theta))$$

We want to find θ to maximize $p(\mathbf{x}_i)$ for $i = 1, \ldots, N$.

Training the Model

Let $p(\mathbf{x}|\mathbf{z};\theta)$ be a factorizing Gaussian with parameters given by a MLP.

$$p(\mathbf{x}|\mathbf{z};\theta) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d(\mathbf{z};\theta), \sigma_d^2(\mathbf{z};\theta))$$

We want to find θ to maximize $p(\mathbf{x}_i)$ for $i = 1, \ldots, N$.

Challenges:

- $p(\mathbf{x}_i) = \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$ is intractable.

Training the Model

Let $p(\mathbf{x}|\mathbf{z};\theta)$ be a factorizing Gaussian with parameters given by a MLP.

$$p(\mathbf{x}|\mathbf{z};\theta) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d(\mathbf{z};\theta), \sigma_d^2(\mathbf{z};\theta))$$

We want to find θ to maximize $p(\mathbf{x}_i)$ for $i = 1, \ldots, N$.

Challenges:

- $p(\mathbf{x}_i) = \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$ is intractable.
- $p(\mathbf{x}_i) = \frac{1}{n}\sum_{j=1}^{n} p(\mathbf{x}_i|\mathbf{z}_j)$, $\mathbf{z}_j \sim p(\mathbf{z})$ demands extremely large n.

Training the Model

Let $p(\mathbf{x}|\mathbf{z};\theta)$ be a factorizing Gaussian with parameters given by a MLP.

$$p(\mathbf{x}|\mathbf{z};\theta) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d(\mathbf{z};\theta), \sigma_d^2(\mathbf{z};\theta))$$

We want to find θ to maximize $p(\mathbf{x}_i)$ for $i = 1, \ldots, N$.

Challenges:

- $p(\mathbf{x}_i) = \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$ is intractable.
- $p(\mathbf{x}_i) = \frac{1}{n}\sum_{j=1}^{n} p(\mathbf{x}_i|\mathbf{z}_j)$, $\mathbf{z}_j \sim p(\mathbf{z})$ demands extremely large n.

The **Variational Autoencoder** (Kingma and Welling, 2014) solves this:

Training the Model

Let $p(\mathbf{x}|\mathbf{z};\theta)$ be a factorizing Gaussian with parameters given by a MLP.

$$p(\mathbf{x}|\mathbf{z};\theta) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d(\mathbf{z};\theta), \sigma_d^2(\mathbf{z};\theta))$$

We want to find θ to maximize $p(\mathbf{x}_i)$ for $i = 1, \ldots, N$.

Challenges:

- $p(\mathbf{x}_i) = \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$ is intractable.
- $p(\mathbf{x}_i) = \frac{1}{n}\sum_{j=1}^{n} p(\mathbf{x}_i|\mathbf{z}_j)$, $\mathbf{z}_j \sim p(\mathbf{z})$ demands extremely large n.

The **Variational Autoencoder** (Kingma and Welling, 2014) solves this:

- Samples values of $\mathbf{z}$ that are **likely to have produced $\mathbf{x}_i$**.

Training the Model

Let $p(\mathbf{x}|\mathbf{z};\theta)$ be a factorizing Gaussian with parameters given by a MLP.

$$p(\mathbf{x}|\mathbf{z};\theta) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d(\mathbf{z};\theta), \sigma_d^2(\mathbf{z};\theta))$$

We want to find θ to maximize $p(\mathbf{x}_i)$ for $i = 1, \ldots, N$.

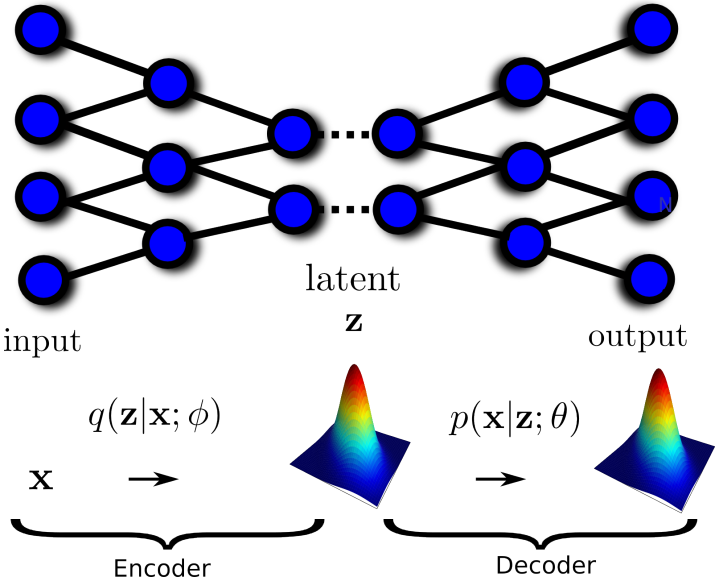
Challenges:

- $p(\mathbf{x}_i) = \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$ is intractable.
- $p(\mathbf{x}_i) = \frac{1}{n}\sum_{j=1}^{n} p(\mathbf{x}_i|\mathbf{z}_j)$, $\mathbf{z}_j \sim p(\mathbf{z})$ demands extremely large n.

The **Variational Autoencoder** (Kingma and Welling, 2014) solves this:

- Samples values of $\mathbf{z}$ that are **likely to have produced $\mathbf{x}_i$**.
- Adds a **recognition network** $q(\mathbf{z}|\mathbf{x};\phi)$ that approximates $p(\mathbf{z}|\mathbf{x})$.

Variational Autoencoder



VAE: Training the Networks

The VAE maximizes $\log p(\mathbf{x}_i)$ w.r.t θ in an **approximate way**:

VAE: Training the Networks

The VAE maximizes $\log p(\mathbf{x}_i)$ w.r.t θ in an **approximate way**:

$$\begin{aligned}
\log p(\mathbf{x}_i) &= \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})\frac{q(\mathbf{z}|\mathbf{x}_i;\phi)}{q(\mathbf{z}|\mathbf{x}_i;\phi)}d\mathbf{z} \\
&= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \\
&\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\log \frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}[\log p(\mathbf{x}_i|\mathbf{z};\theta)] - \mathrm{KL}\left(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z})\right) \equiv \mathcal{L}(\mathbf{x}_i;\theta,\phi)
\end{aligned}$$

VAE: Training the Networks

The VAE maximizes $\log p(\mathbf{x}_i)$ w.r.t θ in an **approximate way**:

$$\begin{aligned}\log p(\mathbf{x}_i) &= \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})\frac{q(\mathbf{z}|\mathbf{x}_i;\phi)}{q(\mathbf{z}|\mathbf{x}_i;\phi)}d\mathbf{z} \\ &= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \\ &\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\log \frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\log p(\mathbf{x}_i|\mathbf{z};\theta)\right] - \mathrm{KL}\left(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z})\right) \equiv \mathcal{L}(\mathbf{x}_i;\theta,\phi)\end{aligned}$$

The **difference** is the KL divergence between $q(\mathbf{z}|\mathbf{x}_i;\phi)$ and $p(\mathbf{z}|\mathbf{x}_i)$.

$$\log p(\mathbf{x}_i) - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z}|\mathbf{x}_i)) = \mathcal{L}(\mathbf{x}_i;\theta,\phi)$$

VAE: Training the Networks

The VAE maximizes $\log p(\mathbf{x}_i)$ w.r.t θ in an **approximate way**:

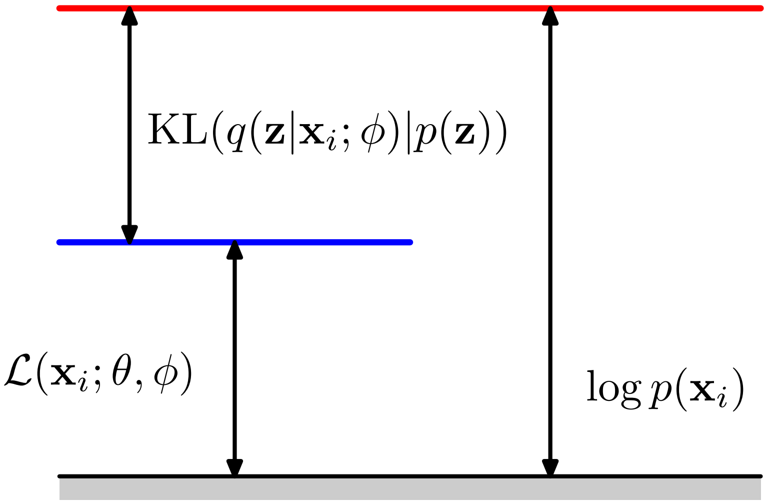
$$
\begin{aligned}
\log p(\mathbf{x}_i) &= \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})\frac{q(\mathbf{z}|\mathbf{x}_i;\phi)}{q(\mathbf{z}|\mathbf{x}_i;\phi)}d\mathbf{z} \\
&= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \\
&\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\log \frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}[\log p(\mathbf{x}_i|\mathbf{z};\theta)] - \mathrm{KL}\left(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z})\right) \equiv \mathcal{L}(\mathbf{x}_i;\theta,\phi)
\end{aligned}
$$

The **difference** is the KL divergence between $q(\mathbf{z}|\mathbf{x}_i;\phi)$ and $p(\mathbf{z}|\mathbf{x}_i)$.

$$
\log p(\mathbf{x}_i) - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z}|\mathbf{x}_i)) = \mathcal{L}(\mathbf{x}_i;\theta,\phi)
$$

Maximizing $\mathcal{L}(\mathbf{x}_i;\theta,\phi)$ w.r.t ϕ makes $\mathrm{KL}(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z}|\mathbf{x}_i))$ very small, and maximizing w.r.t. θ should improve $\log p(\mathbf{x}_i)$.

VAE: Training the Networks



VAE: Training the Networks

Given a dataset $\{\mathbf{x}_i\}_{i=1}^{N}$ the objective is:

$$\sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i; \theta, \phi) = \sum_{i=1}^{N} \left(\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)} \left[\log p(\mathbf{x}_i|\mathbf{z};\theta) \right] - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z})) \right)$$

VAE: Training the Networks

Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$ the objective is:

$$\sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i; \theta, \phi) = \sum_{i=1}^{N} \left(\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)} [\log p(\mathbf{x}_i|\mathbf{z};\theta)] - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z})) \right)$$

Each term $\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)} [\log p(\mathbf{x}_i|\mathbf{z};\theta)]$ is approximated using **black-box** VI:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)} [\log p(\mathbf{x}_i|\mathbf{z};\theta)] \approx \frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{x}_i|\mathbf{z}^{(m)};\theta), \quad \mathbf{z}^{(m)} \sim q(\mathbf{z}|\mathbf{x}_i;\phi)$$

VAE: Training the Networks

Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$ the objective is:

$$\sum_{i=1}^N \mathcal{L}(\mathbf{x}_i;\theta,\phi) = \sum_{i=1}^N \left(\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\log p(\mathbf{x}_i|\mathbf{z};\theta)\right] - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}_i;\phi)|p(\mathbf{z}))\right)$$

Each term $\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\log p(\mathbf{x}_i|\mathbf{z};\theta)\right]$ is approximated using **black-box** VI:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\log p(\mathbf{x}_i|\mathbf{z};\theta)\right] \approx \frac{1}{M}\sum_{m=1}^M \log p(\mathbf{x}_i|\mathbf{z}^{(m)};\theta), \quad \mathbf{z}^{(m)} \sim q(\mathbf{z}|\mathbf{x}_i;\phi)$$

The **reparametrization trick** allows to compute gradients w.r.t ϕ:

$$\mathbf{z}^{(m)} = \mathbf{L}(\mathbf{x}_i;\phi)^{\mathsf{T}}\boldsymbol{\epsilon} + \boldsymbol{\mu}(\mathbf{x}_i;\phi), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$$

VAE: Training the Networks

Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$ the objective is:

$$\sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i; \theta, \phi) = \sum_{i=1}^{N} \left(\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)} \left[\log p(\mathbf{x}_i|\mathbf{z};\theta) \right] - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}_i;\phi) | p(\mathbf{z})) \right)$$

Each term $\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)} \left[\log p(\mathbf{x}_i|\mathbf{z};\theta) \right]$ is approximated using **black-box** VI:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)} \left[\log p(\mathbf{x}_i|\mathbf{z};\theta) \right] \approx \frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{x}_i|\mathbf{z}^{(m)};\theta), \quad \mathbf{z}^{(m)} \sim q(\mathbf{z}|\mathbf{x}_i;\phi)$$

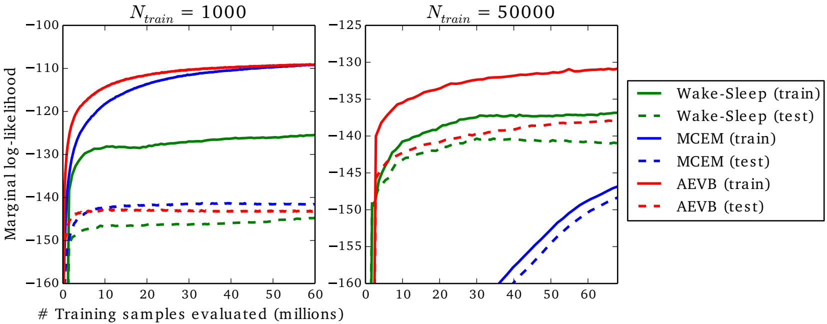
The **reparametrization trick** allows to compute gradients w.r.t ϕ:

$$\mathbf{z}^{(m)} = \mathbf{L}(\mathbf{x}_i;\phi)^{\mathrm{T}} \boldsymbol{\epsilon} + \boldsymbol{\mu}(\mathbf{x}_i;\phi), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

We can use minibatches and stochastic gradients for training!
Furthermore, all MLP operations can be done in the GPU.

Results on the MNIST Dataset

100 hidden units in the MLP and 3 latent variables:

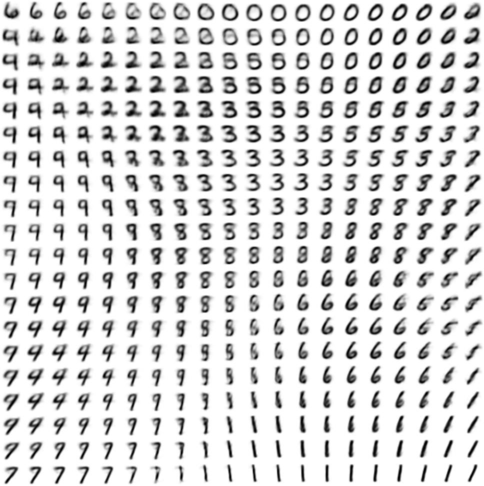


(Kingma and Welling, 2014)

2D Manifolds Learned by the VAE

The **z**'s are transformed using the inverse CDF of the standard Gaussian.

(a) Learned Frey Face manifold



(b) Learned MNIST manifold

(Kingma and Welling, 2014)

Generated samples from the MNIST

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

(Kingma and Welling, 2014)

Importance Weighted Autoencoder

Improves the VAE by considering a **tighter lower bound** on $p(\mathbf{x}_i)$.

Importance Weighted Autoencoder

Improves the VAE by considering a **tighter lower bound** on $p(\mathbf{x}_i)$.

Consider an **importance sampling** estimate of $p(\mathbf{x}_i)$:

Importance Weighted Autoencoder

Improves the VAE by considering a **tighter lower bound** on $p(\mathbf{x}_i)$.

Consider an **importance sampling** estimate of $p(\mathbf{x}_i)$:

$$\log p(\mathbf{x}_i) = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})\frac{q(\mathbf{z}|\mathbf{x}_i;\phi)}{q(\mathbf{z}|\mathbf{x}_i;\phi)}d\mathbf{z}$$

$$= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \approx \log \frac{1}{k}\sum_{m=1}^{k}\frac{p(\mathbf{x}_i|\mathbf{z}^{(m)};\theta)p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)}|\mathbf{x}_i;\phi)}$$

Importance Weighted Autoencoder

Improves the VAE by considering a **tighter lower bound** on $p(\mathbf{x}_i)$.

Consider an **importance sampling** estimate of $p(\mathbf{x}_i)$:

$$\log p(\mathbf{x}_i) = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})\frac{q(\mathbf{z}|\mathbf{x}_i;\phi)}{q(\mathbf{z}|\mathbf{x}_i;\phi)}d\mathbf{z}$$

$$= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \approx \log \frac{1}{k}\sum_{m=1}^{k}\frac{p(\mathbf{x}_i|\mathbf{z}^{(m)};\theta)p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)}|\mathbf{x}_i;\phi)}$$

On expectation that estimate is a **lower bound** on $\log p(\mathbf{x}_i)$:

$$\mathcal{L}_k(\mathbf{x}_i;\theta,\phi) = \mathbb{E}\left[\log \frac{1}{k}\sum_{m=1}^{k} w_m\right] \le \log \mathbb{E}\left[\frac{1}{k}\sum_{m=1}^{k} w_m\right] = \log p(\mathbf{x}_i)$$

Importance Weighted Autoencoder

Improves the VAE by considering a **tighter lower bound** on $p(\mathbf{x}_i)$.

Consider an **importance sampling** estimate of $p(\mathbf{x}_i)$:

$$\log p(\mathbf{x}_i) = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \log \int p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})\frac{q(\mathbf{z}|\mathbf{x}_i;\phi)}{q(\mathbf{z}|\mathbf{x}_i;\phi)}d\mathbf{z}$$

$$= \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\phi)}\left[\frac{p(\mathbf{x}_i|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_i;\phi)}\right] \approx \log \frac{1}{k}\sum_{m=1}^{k} \frac{p(\mathbf{x}_i|\mathbf{z}^{(m)};\theta)p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)}|\mathbf{x}_i;\phi)}$$

On expectation that estimate is a **lower bound** on $\log p(\mathbf{x}_i)$:

$$\mathcal{L}_k(\mathbf{x}_i;\theta,\phi) = \mathbb{E}\left[\log \frac{1}{k}\sum_{m=1}^{k} w_m\right] \leq \log \mathbb{E}\left[\frac{1}{k}\sum_{m=1}^{k} w_m\right] = \log p(\mathbf{x}_i)$$

If $k = 1$ we obtain the VAE. $k > 1$ can only improve the bound.
Optimization is done as in the VAE.

(Burda *et al.*, 2016)

Experimental Results

		MNIST				OMNIGLOT			
		VAE		IWAE		VAE		IWAE	
# stoch. layers	k	NLL	active units	NLL	active units	NLL	active units	NLL	active units
1	1	86.76	19	86.76	19	108.11	28	108.11	28
	5	86.47	20	85.54	22	107.62	28	106.12	34
	50	86.35	20	84.78	25	107.80	28	104.67	41
2	1	85.33	16+5	85.33	16+5	107.58	28+4	107.56	30+5
	5	85.01	17+5	83.89	21+5	106.31	30+5	104.79	38+6
	50	84.78	17+5	82.90	26+7	106.30	30+5	103.38	44+7

(Burda *et al.*, 2016)

Uncertainty in the Network Weights

We consider more flexible networks in the IWAE by introducing **uncertainty** in the neural network parameters θ and ϕ.

Uncertainty in the Network Weights

We consider more flexible networks in the IWAE by introducing **uncertainty** in the neural network parameters θ and ϕ.

$$p(\mathbf{x}|\mathbf{z}) = \int p(\mathbf{x}|\mathbf{z};\theta)q(\theta)d\theta\,, \qquad q(\mathbf{z}|\mathbf{x}) = \int q(\mathbf{z}|\mathbf{x};\phi)q(\phi)d\phi\,,$$

Uncertainty in the Network Weights

We consider more flexible networks in the IWAE by introducing **uncertainty** in the neural network parameters θ and ϕ.

$$p(\mathbf{x}|\mathbf{z}) = \int p(\mathbf{x}|\mathbf{z};\theta)q(\theta)d\theta\,, \qquad q(\mathbf{z}|\mathbf{x}) = \int q(\mathbf{z}|\mathbf{x};\phi)q(\phi)d\phi\,,$$

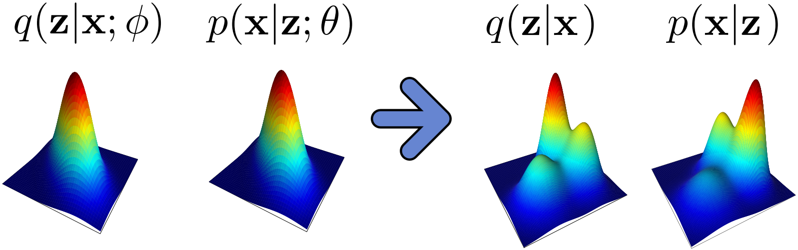
$q(\theta)$ and $q(\phi)$ are Gaussians with parameters $\Omega = \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}^2_\theta, \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}^2_\phi\}$.

Uncertainty in the Network Weights

We consider more flexible networks in the IWAE by introducing **uncertainty** in the neural network parameters θ and ϕ.

$$p(\mathbf{x}|\mathbf{z}) = \int p(\mathbf{x}|\mathbf{z};\theta)q(\theta)d\theta\,, \qquad q(\mathbf{z}|\mathbf{x}) = \int q(\mathbf{z}|\mathbf{x};\phi)q(\phi)d\phi\,,$$

$q(\theta)$ and $q(\phi)$ are Gaussians with parameters $\Omega = \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}^2_\theta, \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}^2_\phi\}$.



Training the Model

Given a dataset $\{\mathbf{x}_i\}_{i=1}^{N}$ the **objective** is:

$$\sum_{i=1}^{N} \mathcal{L}_k(\mathbf{x}_i; \Omega) = \sum_{i=1}^{N} \mathbb{E}\left[\log \sum_{m=1}^{k} \frac{p(\mathbf{x}_i|\mathbf{z}^{(m)}; \theta^{(m)})p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)}|\mathbf{x}_i; \phi^{(m)})}\right]$$

where $\mathbf{z}^{(m)}$, $\theta^{(m)}$ and $\phi^{(m)}$ are sampled from $q(\mathbf{z}|\mathbf{x}_i; \phi^{(m)})$, $q(\theta)$ and $q(\phi)$.

Training the Model

Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$ the **objective** is:

$$\sum_{i=1}^{N} \mathcal{L}_k(\mathbf{x}_i; \Omega) = \sum_{i=1}^{N} \mathbb{E}\left[\log \sum_{m=1}^{k} \frac{p(\mathbf{x}_i|\mathbf{z}^{(m)}; \theta^{(m)})p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)}|\mathbf{x}_i; \phi^{(m)})}\right]$$

where $\mathbf{z}^{(m)}$, $\theta^{(m)}$ and $\phi^{(m)}$ are sampled from $q(\mathbf{z}|\mathbf{x}_i; \phi^{(m)})$, $q(\theta)$ and $q(\phi)$.

The expectation $\mathbb{E}[\cdot]$ is approximated by one Monte Carlo sample.

Training the Model

Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$ the **objective** is:

$$\sum_{i=1}^{N} \mathcal{L}_k(\mathbf{x}_i;\Omega) = \sum_{i=1}^{N} \mathbb{E}\left[\log \sum_{m=1}^{k} \frac{p(\mathbf{x}_i|\mathbf{z}^{(m)};\theta^{(m)})p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)}|\mathbf{x}_i;\phi^{(m)})}\right]$$

where $\mathbf{z}^{(m)}$, $\theta^{(m)}$ and $\phi^{(m)}$ are sampled from $q(\mathbf{z}|\mathbf{x}_i;\phi^{(m)})$, $q(\theta)$ and $q(\phi)$.

The expectation $\mathbb{E}[\cdot]$ is approximated by one Monte Carlo sample.

We use stochastic gradients and the **local reparametrization trick**:

Training the Model

Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$ the **objective** is:

$$\sum_{i=1}^{N} \mathcal{L}_k(\mathbf{x}_i; \Omega) = \sum_{i=1}^{N} \mathbb{E}\left[\log \sum_{m=1}^{k} \frac{p(\mathbf{x}_i|\mathbf{z}^{(m)}; \theta^{(m)})p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)}|\mathbf{x}_i; \phi^{(m)})}\right]$$

where $\mathbf{z}^{(m)}$, $\theta^{(m)}$ and $\phi^{(m)}$ are sampled from $q(\mathbf{z}|\mathbf{x}_i; \phi^{(m)})$, $q(\theta)$ and $q(\phi)$.

The expectation $\mathbb{E}[\cdot]$ is approximated by one Monte Carlo sample.

We use stochastic gradients and the **local reparametrization trick**:

1. Sample the NN activations $\mathbf{A} = \mathbf{XW} + \mathbf{b1}^\mathsf{T}$, which are Gaussian.

Training the Model

Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$ the **objective** is:

$$\sum_{i=1}^{N} \mathcal{L}_k(\mathbf{x}_i; \Omega) = \sum_{i=1}^{N} \mathbb{E}\left[\log \sum_{m=1}^{k} \frac{p(\mathbf{x}_i|\mathbf{z}^{(m)}; \theta^{(m)}) p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)}|\mathbf{x}_i; \phi^{(m)})}\right]$$

where $\mathbf{z}^{(m)}$, $\theta^{(m)}$ and $\phi^{(m)}$ are sampled from $q(\mathbf{z}|\mathbf{x}_i; \phi^{(m)})$, $q(\theta)$ and $q(\phi)$.

The expectation $\mathbb{E}[\cdot]$ is approximated by one Monte Carlo sample.

We use stochastic gradients and the **local reparametrization trick**:

1. Sample the NN activations $\mathbf{A} = \mathbf{X}\mathbf{W} + \mathbf{b}\mathbf{1}^\mathsf{T}$, which are Gaussian.
2. Instead of sampling $M \times H \times D$ variables, we sample $M \times H$.

(Kingma *et al.*, 2015)

Experimental Results: MNIST and Omniglot

- We consider 1-layer MLP with 400 units and 40 latent variables.
- We compare with a model that considers uncertainty only in ϕ.
- We set the number of importance samples $k = 25$.

Average test log-likelihood for each method.

Dataset	**IWAE**	**IWAEU**	**IWAEU$_{\text{rec}}$**
MNIST	-95.182±0.022	**-94.346±0.025**	-94.709±0.025
Omniglot	-118.771±0.035	**-118.540±0.049**	-118.647±0.031

Conclusions and Future Work

Conclusions:

1. The **VAE** and the **IWAE** are powerful generative models based on latent variables for unsupervised machine learning.

Conclusions and Future Work

Conclusions:

1. The **VAE** and the **IWAE** are powerful generative models based on latent variables for unsupervised machine learning.

2. The performance of the IWAE can be **improved** by considering **random neural network weights** in both networks.

Conclusions and Future Work

Conclusions:

1. The **VAE** and the **IWAE** are powerful generative models based on latent variables for unsupervised machine learning.
2. The performance of the IWAE can be **improved** by considering **random neural network weights** in both networks.

Future Work:

1. Carry out extra experiments to explore if the gains are also obtained with **bigger and deeper** neural networks.

Conclusions and Future Work

Conclusions:

1. The **VAE** and the **IWAE** are powerful generative models based on latent variables for unsupervised machine learning.

2. The performance of the IWAE can be **improved** by considering **random neural network weights** in both networks.

Future Work:

1. Carry out extra experiments to explore if the gains are also obtained with **bigger and deeper** neural networks.

2. Combine with **black-box-alpha** for training and explore **other models** (e.g., ladder variational autoencoders).

References

- D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In International Conference on Learning Representations, 2014.
- Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In International Conference on Learning Representations, 2016.
- D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In Advances in Neural Information Processing Systems. 2015.
- D. P. Kingma and J. L Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations, 2015.
- C. K. Sonderby, T. Raiko, L. Maaloe, S. K. Sonderby, and O. Winther. Ladder variational autoencoders. 2016. arXiv:1602.02282.