VAE

A Detailed Introduction to Variational Autoencoder

Zhenya Liu

Overview

1. Background

2. Variational Autoencoder

3. Experiments

Autoencoder

Autoencoder is a neural network designed to learn an identity function in an unsupervised way to reconstruct the original input while compressing the data in the process so as to discover a more efficient and compressed representation.

This idea was originated from 1980s, and formally proposed by Hinton & Salakhutdinov,2006 [1].

Use neural networks to define: encoder g_ϕ and decoder f_θ.

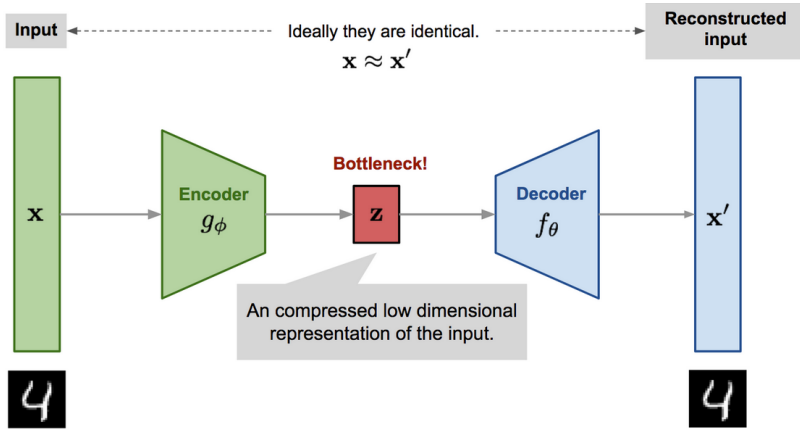


Figure: Illustration of autoencoder model architecture[4]

Autoencoder

Training Objective:

$$L_{\text{AE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)} - f_\theta \left(g_\phi \left(\mathbf{x}^{(i)} \right) \right) \right)^2$$

Autoencoder

Unavoidable problem – Overfitting.
Several works on autoencoders were invented to train autoencoders with less overfitting and more robustness.
For instance, in **Denoising Autoencoder**(Vincent et al. 2008), the inputs are partially corrupted by masks or noises

$$\tilde{\mathbf{x}}^{(i)} \sim \mathcal{M}_{\mathcal{D}}\left(\tilde{\mathbf{x}}^{(i)} \mid \mathbf{x}^{(i)}\right)$$

$$L_{\mathrm{DAE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)} - f_\theta\left(g_\phi\left(\tilde{\mathbf{x}}^{(i)}\right)\right)\right)^2$$

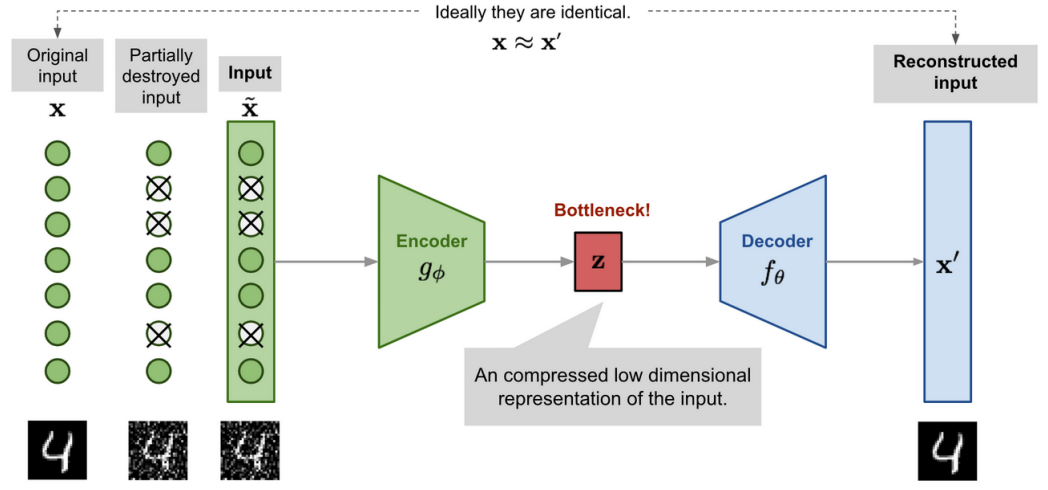


Figure: Illustration of denoising autoencoder model architecture[4]

EM Algorithm

Expectation Maximization (EM) algorithm is an iterative method to find local maximum likelihood estimates (MLE) of parameters in statistical models, where the model depends on unobserved latent variables.

EM Algorithm

Problem Setup

Given one model parameterized by $\boldsymbol{\theta}$, we aim to find $\boldsymbol{\theta}^*$ maximize the marginal likelihood of observed data $\mathbf{X}$, *i.e* MLE of $P(\mathbf{X}|\boldsymbol{\theta})$. We also have unobserved latent variable $\mathbf{Z}$ such that $P(\mathbf{Z})$ is unknown, but $P(\mathbf{Z}|\boldsymbol{\theta}), P(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta})$ is known.

$$L(\boldsymbol{\theta}; \mathbf{X}) = p(\mathbf{X} \mid \boldsymbol{\theta}) = \int p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) d\mathbf{Z} = \int p(\mathbf{X} \mid \mathbf{Z}, \boldsymbol{\theta}) p(\mathbf{Z} \mid \boldsymbol{\theta}) d\mathbf{Z}$$

In some cases, the above integral can be calculated, so we get one function over $\boldsymbol{\theta}$. However, we always cannot find the closed form solution on $\boldsymbol{\theta}^*$ because solutions always have circular dependencies. Moreover, this integral is always non-concave, so we can only expect to find a local maximum.

EM Algorithm

Expectation step(E step):

$$Q\left(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}\right) = \mathrm{E}_{\mathbf{Z}\sim p\left(\cdot|\mathbf{X},\boldsymbol{\theta}^{(t)}\right)}[\log p(\mathbf{X},\mathbf{Z})|\boldsymbol{\theta})]$$

Maximization step(M step):

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q\left(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}\right)$$

Equivalently,

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \mathrm{E}_{\mathbf{Z}\sim p\left(\cdot|\mathbf{X},\boldsymbol{\theta}^{(t)}\right)}[\log p(\mathbf{X},\mathbf{Z})|\boldsymbol{\theta})]$$

EM Algorithm

Example: Gaussian Mixture Model

Given observed set of datapoints $\{X_i\}$, and set of unobserved latent variables $\{Z_i\}$

$$X_i \mid (Z_i = 1) \sim \mathcal{N}_d(\mu_1, \Sigma_1) \text{ and } X_i \mid (Z_i = 2) \sim \mathcal{N}_d(\mu_2, \Sigma_2),$$

where

$$\mathrm{P}(Z_i = 1) = \tau_1 \text{ and } \mathrm{P}(Z_i = 2) = \tau_2 = 1 - \tau_1.$$

We can use EM Algorithm to find

$$\theta = (\boldsymbol{\tau}, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2)$$

Variational Inference

Problem Setup

Given prior $P(\mathbf{Z})$ and likelihood $P(\mathbf{X} \mid \mathbf{Z})$, aim to find the posterior distribution

$$P(\mathbf{Z} \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Z})}{P(\mathbf{X})} = \frac{P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Z})}{\int_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z}')\, d\mathbf{Z}'}$$

Generally, the space of $\mathbf{Z}$ is large so that the integral part is intractable, so we need to find $Q(\mathbf{Z}) \approx P(\mathbf{Z} \mid \mathbf{X})$.

Question:
how do we compare two distributions? how to find a tractable formula on $Q(\mathbf{Z})$ to minimize the "distance" between them?

KL Divergence

Kullback–Leibler Divergence

For discrete probability distributions P and Q defined on the same sample space, $\mathcal{X}$,

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right)$$

KL divergence $D_{\mathrm{KL}}(P\|Q)$ measures how much information is lost if the distribution Q is used to represent P.

KL Divergence

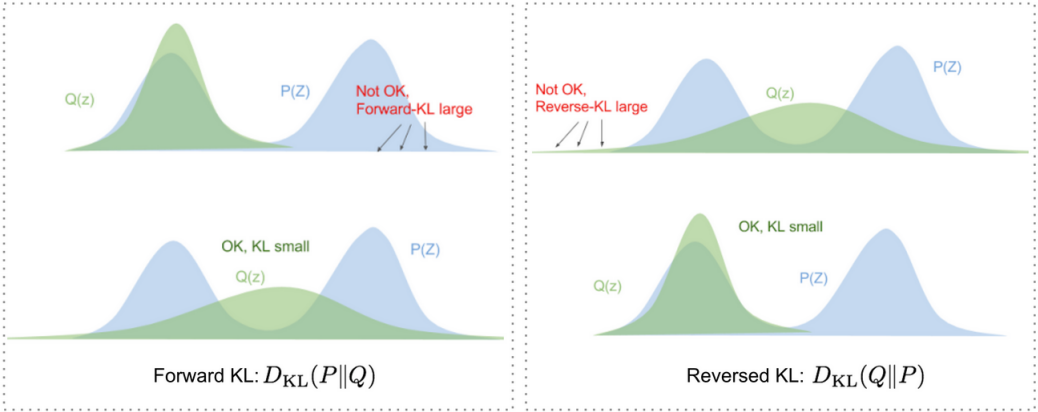


Figure: Forward and reversed KL divergence have different demands on how to match two distributions[4].

KL Divergence

$$
\begin{aligned}
& D_{\mathrm{KL}}(Q(\mathbf{Z}) \| P(\mathbf{Z}|\mathbf{X})) \\
&= \sum_{\mathbf{Z}} Q(\mathbf{Z}) \left[\log \frac{Q(\mathbf{Z})}{P(\mathbf{Z}, \mathbf{X})} + \log P(\mathbf{X}) \right] \\
&= \sum_{\mathbf{Z}} Q(\mathbf{Z}) [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \sum_{\mathbf{Z}} Q(\mathbf{Z}) [\log P(\mathbf{X})] \\
&= \mathbb{E}_{\mathbf{Q}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X})
\end{aligned}
$$

KL Divergence

$$\begin{aligned}\log P(\mathbf{X}) &= D_{\mathrm{KL}}(Q(\mathbf{Z})\|P(\mathbf{Z}|\mathbf{X})) - \mathbb{E}_{\mathbf{Q}}[\log Q(\mathbf{Z}) - \log P(\mathbf{Z},\mathbf{X})] \\ &= D_{\mathrm{KL}}(Q(\mathbf{Z})\|P(\mathbf{Z}|\mathbf{X})) + \mathcal{L}(Q)\end{aligned}$$

Here $\mathcal{L}(Q) = -\mathbb{E}_{\mathbf{Q}}[\log Q(\mathbf{Z}) - \log P(\mathbf{Z},\mathbf{X})]$ is known as the **Evidence Lower Bound** (ELBO) of $\log P(\mathbf{X})$.

Most importantly, $\mathcal{L}(Q)$ is tractable and optimizable.

KL Divergence

Here is another way to get ELBO by using *Jensen's Inequality:*

$$
\begin{aligned}
\log p(\mathbf{X}) &= \log \int_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}) \\
&= \log \int_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}) \frac{Q(\mathbf{Z})}{Q(\mathbf{Z})} \\
&= \log \left(\mathrm{E}_Q \left[\frac{p(\mathbf{X}, \mathbf{Z})}{Q(\mathbf{Z})} \right] \right) \\
&\geq \mathrm{E}_Q[\log p(\mathbf{X}, \mathbf{Z})] - \mathrm{E}_Q[\log Q(\mathbf{Z})] \\
&= \mathcal{L}(Q)
\end{aligned}
$$

KL Divergence

Back to what we get

$$\log P(\mathbf{X}) = D_{\mathrm{KL}}(Q(\mathbf{Z}) \| P(\mathbf{Z}|\mathbf{X})) + \mathcal{L}(Q)$$

So

$$\mathcal{L}(Q) = \log P(\mathbf{X}) - D_{\mathrm{KL}}(Q(\mathbf{Z}) \| P(\mathbf{Z}|\mathbf{X}))$$

Maximize $\mathcal{L}(Q) \implies$ maximize $\log P(\mathbf{X})$ and minimize $D_{\mathrm{KL}}(Q(\mathbf{Z}) \| P(\mathbf{Z}|\mathbf{X}))$.

ELBO and EM Algorithm

Based on $\mathcal{L}(\mathrm{Q})$, we can prove the effectiveness of EM Algorithm.
Since

$$\log P(\mathbf{X}) = \mathcal{L}(Q) + D_{\mathrm{KL}}(Q(\mathbf{Z}) \| P(\mathbf{Z} \mid \mathbf{X}))$$

When $Q(\mathbf{Z}) = P(\mathbf{Z} \mid \mathbf{X})$,

$$\log P(\mathbf{X} \mid \boldsymbol{\theta}) = \underbrace{\mathbb{E}_{P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta})}[\log P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})]}_{\text{EM step maximizes it}} - \mathbb{E}_{P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta})}[\log P(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})]$$

For one iteration of EM Algorithm, let $\boldsymbol{\theta}^{(t)}$ be the initial value, then we can also get

$$\log P(\mathbf{X} \mid \boldsymbol{\theta}) = \underbrace{\mathbb{E}_{P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)})}[\log P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})]}_{H(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})} \underbrace{- \mathbb{E}_{P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)})}[\log P(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})]}_{I(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})}$$

Question: why?

ELBO and EM Algorithm

During this iteration, we get $\boldsymbol{\theta}^{(t+1)}$, and

$$\log p\left(\mathbf{X} \mid \boldsymbol{\theta}^{(t+1)}\right) = H\left(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}\right) + I\left(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}\right)$$

By *Gibb's Inequality*,

$$\underbrace{I\left(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}\right)}_{\text{cross entropy}} - \underbrace{I\left(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)}\right)}_{\text{entropy}} \geq 0$$

ELBO and EM Algorithm

Thus, for each EM update,

$$
\begin{aligned}
&\log p(\mathbf{x} \mid \boldsymbol{\theta}^{(t+1)}) - \log p\left(\mathbf{x} \mid \boldsymbol{\theta}^{(t)}\right) \\
&= H\left(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}\right) - H\left(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)}\right) + I\left(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}\right) - I\left(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)}\right) \\
&\geq H\left(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}\right) - H\left(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)}\right)
\end{aligned}
$$

This shows that updating $\boldsymbol{\theta}$ to improve $H\left(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}\right)$ causes $\log p(\mathbf{X}|\boldsymbol{\theta})$ to improve at least as much.

Variational Inference

We have decided the optimization objective $\mathcal{L}(Q)$, then we need to decide how to represent $Q(\mathbf{Z})$. We expect to give $Q(\mathbf{Z})$ high flexibility and tractable computability over expectation.
In variational inference, we pick a family of distributions over the latent variables with its own variational parameters, $i.e$

$$Q(\mathbf{Z}) = q\left(z_{1:m}|\nu\right)$$

Mean Field Variational Inference

Assume that the variational family factorizes

$$q(z_1, \ldots, z_m) = \prod_{j=1}^{m} q(z_j)$$

Here we assume all latent variables are independent. This also restricts our approximation to the true posterior distribution because latent variables are always dependent, *i.e* latent variables in Gaussian mixture models.

Mean Field Variational Inference

Define $\mathbf{X} = x_{1:n}$, and choose one z_j from $z_{1:m}$. Refer $-j$ to $\{1, ..., m\} \setminus \{j\}$,

$$E_{q(z_{1:m})} \left[\log P\left(z_{1:m}, x_{1:n}\right)\right] = \int_{z_j} q_j\left(z_j\right) E_{-j}\left[\log P(x_{1:n}, z_{-j})\right] dz_j$$

Evaluate the expectation

$$\mathrm{E}_{q(z_{1:m})}\left[\log q\left(z_{1:m}\right)\right] = \sum_{j=1}^{m} \mathrm{E}_j\left[\log q\left(z_j\right)\right] + C$$

where E_j represents the expectation with respect to $q(z_j)$, and C is some terms irrelevant with $q(z_j)$.

Mean Field Variational Inference

Now calculate $\mathcal{L}(Q)$

$$\begin{aligned}\mathcal{L} &= E_{q(z_{1:m})}\left[\log P\left(z_{1:m}, x_{1:n}\right) - \log q\left(z_{1:m}\right)\right] \\ &= \int_{z_j} q_j\left(z_j\right) E_{-j}\left[\log P(x_{1:n}, z_{-j})\right] dz_j - \sum_{j=1}^{m} \mathrm{E}_j\left[\log q\left(z_j\right)\right] - C\end{aligned}$$

From now on, we treat $\mathcal{L}$ as one **functional** of $q(z_j)$ while fixing all other $q(z_{-j})$ and we want to find $q^*(z_j)$ which maximizes $\mathcal{L}$.

How to find such $q^*(z_j)$? Use methods in **Calculus of Variation**.

Calculus of Variation

Theorem (Euler Lagrange Equation)

Given a functional $J(f)$ defined on all functions $f \in C^2[a,b]$ such that $f(a) = A, f(b) = B$,

$$J(f) = \int_a^b L\left(x, f, f'\right) \mathrm{d}x$$

then if $J(f)$ is a local extremum, then f satisfies

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial L}{\partial f'}\right) - \frac{\partial L}{\partial f} = 0$$

Calculus of Variation

Example

$$\mathrm{s} = \int_a^b \sqrt{\mathrm{d}x^2 + \mathrm{d}y^2} = \int_a^b \sqrt{1 + y'^2}\ \mathrm{d}x$$

Let $L\left(x, y, y'\right) = \sqrt{1 + y'^2}$,

$$\frac{\partial L\left(x, y, y'\right)}{\partial y'} = \frac{y'}{\sqrt{1 + y'^2}} \quad \text{and} \quad \frac{\partial L\left(x, y, y'\right)}{\partial y} = 0$$

By **Euler Lagrange Equation**,

$$\frac{\mathrm{d}}{\mathrm{d}x} \frac{y'(x)}{\sqrt{1 + \left(y'(x)\right)^2}} = 0, \frac{y'(x)}{\sqrt{1 + \left(y'(x)\right)^2}} = C$$

$$\Rightarrow y'(x) = \frac{C}{\sqrt{1 - C^2}} = A$$

$$\Rightarrow y(x) = Ax + B$$

Mean Field Variational Inference

Now we drop out all irrelevant terms with $q(z_j)$ in $\mathcal{L}$, and we get $\mathcal{L}_j$,

$$\begin{aligned}\mathcal{L}_j &= \mathrm{E}_{-j}\left[\log p\left(z_j \mid z_{-j}, x_{1:n}\right)\right] - \mathrm{E}_j\left[\log q\left(z_j\right)\right] \\ &= \int q\left(z_j\right) \mathrm{E}_{-j}\left[\log p\left(z_j \mid z_{-j}, x_{1:n}\right)\right] - q\left(z_j\right) \log q\left(z_j\right) dz_j\end{aligned}$$

Define L be the inner part of above integral, by **Euler Lagrange Equation**,

$$\frac{dL}{dq\left(z_j\right)} = \mathrm{E}_{-j}\left[\log p\left(z_j \mid z_{-j}, x_{1:n}\right)\right] - \log q\left(z_j\right) = 0$$

Then we get expected $q^*(z_j)$ in **uncontrained** case.

Mean Field Variational Inference

Note that since $q(z_j)$ is one probability distribution, so there is one constrain functional $\int q(z_j)dz_j = 1$. We use **Lagrange Multiplier** to handle this situation. In this case, the solution $q(z_j)$ must satisfy

$$\frac{dL}{dq\left(z_j\right)} - \lambda \cdot \frac{dq(z_j)}{dq(z_j)} = 0$$

so

$$\log q_j\left(z_j\right) = E_{-j}\left[\log P\left(X, z_j, z_{-j}\right)\right] + \lambda$$

$$q^*\left(z_j\right) = \exp\left\{\mathrm{E}_{-j}\left[\log p\left(z_j, z_{-j}, x_{1:n}\right)\right]\right\} + \exp(\lambda)$$

where $\exp(\lambda)$ can be figured out when normalizing $q^*\left(z_j\right)$.

Mean Field Variational Inference

So far we only consider maximize $\mathcal{L}$ on one coordinate $q(z_j)$ while fixing other coordinates. In practice, this is the **coordinate ascend** step of the algorithm.

We initialize all $q(z_{1:m})$, and iteratively update each coordinate $q(z_j)$ by previous variational method. Finally we will converge to one local extreme $\mathcal{L}$.

This method is called **Mean Field Variational Inference**. The meaning of "Mean Field" is by the fact that $q^*(z_j)$ is determined by the expectation over all $q(z_{1:m})$.

Variational Autoencoder

Problem Setting

Given $\mathbf{X} = \left\{\mathbf{x}^{(i)}\right\}_{i=1}^{N}$, we aim to model one distribution P parameterized by $\boldsymbol{\theta}$ with one continuous latent variable $\mathbf{z}$ such that

- $\log p_{\boldsymbol{\theta}}(\mathbf{X})$ is maximized
- $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{X})$ can be efficiently approximated
- $p_{\boldsymbol{\theta}}(\mathbf{X})$ can be efficiently approximated

Variational Autoencoder

The only known variables are $\mathbf{X}$ and the prior $p(\mathbf{z})$, and likelihood $p_{\boldsymbol{\theta}}(\mathbf{X}|\mathbf{z})$ when $\boldsymbol{\theta}$ is known.

EM algorithm does not work because we do not know the posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{X})$.

Mean field variational inference does not work here to approximate the posterior because $\boldsymbol{\theta}$ is unknown, and even it is known, the necessary integral is intractable when $\mathbf{z}$ is continuous.

Question: What should we rely on?

Neural Networks

Variational Autoencoder(VAE)(Kingma & Welling, 2014) is a generative model using neural networks to train on ELBO.

Variational Autoencoder

Construct neural networks $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ (*probabilistic decoder*) for likelihood distribution and $q_{\phi}(\mathbf{z}|\mathbf{x})$ (*probabilistic encoder*) for marginal distribution.

Objection: Maximize $\log p_{\boldsymbol{\theta}}(\mathbf{x})$, and approximate $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ by $q_{\phi}(\mathbf{z}|\mathbf{x})$.

Recall:
Maximize $\mathcal{L}(Q) \Longrightarrow$ maximize $\log P(\mathbf{X})$ and minimize $D_{\mathrm{KL}}(Q(\mathbf{Z})|P(\mathbf{Z}|\mathbf{X}))$.

so the **ELBO** is a sufficient objective for jointly training $\boldsymbol{\theta}, \boldsymbol{\phi}$.

Variational Autoencoder

Marginal log likelihood for individual datapoint
$\log p_{\boldsymbol{\theta}}\left(\mathrm{x}^{(1)}, \cdots, \mathrm{x}^{(N)}\right) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}\left(\mathrm{x}^{(i)}\right)$,

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

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

The KL divergence term on **RHS** can always be analytically calculated, but the expectation term (reconstruction error) must be approximated by samples.

Variational Autoencoder

We cannot directly calculate the gradient of $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)} \mid \mathbf{z}\right)\right]$ with respect to $\boldsymbol{\theta}, \boldsymbol{\phi}$. The usual Monte Carlo gradient estimator for this type of problem is

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

where $\mathbf{z}^{(l)} \sim q_{\phi}\left(\mathbf{z} \mid \mathbf{x}^{(i)}\right)$.

This sampling estimator always exhibits a high variance.

Reparameterization

Under certain mild conditions, for $\widetilde{\mathbf{z}} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$, we can reparameterize it with one differentiable transformation $g_\phi(\epsilon, \mathrm{x})$ and noise variable ϵ,

$$\widetilde{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x}) \quad \text{with} \quad \epsilon \sim p(\epsilon)$$

Then we apply Monte Carlo gradient estimator on sampling ϵ,

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

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

Reparameterization

This reparameterization trick yields Stochastic Gradient Variational Bayes (SGVB) estimator for the **ELBO**.

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

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

Variational Autoencoder

Consider the dataset $\mathbf{X}$ with N samples, we approximate the total **ELBO** loss based on **minibatch** $\mathbf{X}^M == \left\{\mathrm{x}^{(i)}\right\}_{i=1}^{M}$,

$$\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{X}) \simeq \widetilde{\mathcal{L}}^M \left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M\right) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}} \left(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}\right)$$

Variational Autoencoder

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

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

Figure: VAE Algorithm [2]

Variational Autoencoder

Example

Define prior $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$.

Both $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ and $q_{\phi}(\mathbf{z} \mid \mathbf{x})$ are multivariate Gaussian distribution with diagonal covariance, *i.e*

$$\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \log \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, {\boldsymbol{\sigma}_{\boldsymbol{\theta}}}^2 \mathbf{I}\right)$$

$$\log q_{\boldsymbol{\phi}}\left(\mathbf{z} \mid \mathbf{x}\right) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\sigma}_{\boldsymbol{\phi}}^2 \mathbf{I})$$

Variational Autoencoder

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

Since

$$\begin{aligned}\int q_{\boldsymbol{\phi}}(\mathbf{z}) \log q_{\boldsymbol{\phi}}(\mathbf{z}) d\mathbf{z} &= \int \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\sigma}_{\boldsymbol{\phi}}^2 \mathbf{I}\right) \log \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\sigma}_{\boldsymbol{\phi}}^2 \mathbf{I}\right) d\mathbf{z} \\ &= -\frac{J}{2} \log(2\pi) - \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log \sigma_j^2\right)\end{aligned}$$

$$\begin{aligned}-D_{KL}\left(\left(q_{\boldsymbol{\phi}}(\mathbf{z}) \| p_{\boldsymbol{\theta}}(\mathbf{z})\right)\right. &= \int q_{\boldsymbol{\phi}}(\mathbf{z}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})\right) d\mathbf{z} \\ &= \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log\left((\sigma_j)^2\right) - (\mu_j)^2 - (\sigma_j)^2\right)\end{aligned}$$

Variational Autoencoder

We reparameterize $\mathbf{z}^{(i,l)} \sim q_\phi\left(\mathbf{z} \mid \mathbf{x}^{(i)}\right)$ as
$\mathbf{z}^{(i,l)} = g_\phi\left(\mathbf{x}^{(i)}, \boldsymbol{\epsilon}^{(l)}\right) = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)}$ where $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

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

Experiments

Encoder

```
# Network parameters
original_dim = 784
intermediate_dim = 256
latent_dim = 2
# Encoder
inputs = Input(shape=(original_dim,))
h = Dense(intermediate_dim, activation='relu')(inputs)
z_mean = Dense(latent_dim)(h)
z_log_var = Dense(latent_dim)(h)
```

Experiments

Reparameterization trick

```
# Reparameterization trick
def sampling(args):
    z_mean, z_log_var = args
    batch = K.shape(z_mean)[0]
    dim = K.int_shape(z_mean)[1]
    epsilon = K.random_normal(shape=(batch, dim))
    return z_mean + K.exp(0.5 * z_log_var) * epsilon

z = Lambda(sampling, output_shape=(latent_dim,))([z_mean, z_log_var])
```

Experiments

Decoder

```
# Decoder
decoder_h = Dense(intermediate_dim, activation='relu')
decoder_mean = Dense(original_dim, activation='sigmoid')
h_decoded = decoder_h(z)
x_decoded_mean = decoder_mean(h_decoded)
```

Experiments

End-to-end Autoencoder

```
# End-to-end autoencoder
vae = Model(inputs, x_decoded_mean)

# Encoder, from inputs to latent space
encoder = Model(inputs, z_mean)

# Generator, from latent space to reconstructed inputs
decoder_input = Input(shape=(latent_dim,))
_h_decoded = decoder_h(decoder_input)
_x_decoded_mean = decoder_mean(_h_decoded)
generator = Model(decoder_input, _x_decoded_mean)
```

Experiments

Loss function & Train

```
# Loss
xent_loss = original_dim * binary_crossentropy(inputs, x_decoded_mean)
kl_loss = - 0.5 * K.sum(1 + z_log_var - K.square(z_mean) - K.exp(
    z_log_var), axis=-1)
vae_loss = K.mean(xent_loss + kl_loss)

vae.add_loss(vae_loss)
vae.compile(optimizer='rmsprop')

# Train
vae.fit(x_train, epochs=50, batch_size=256, validation_data=(x_test,
    None))
```

VAE implicitly maps one class of number to one neighbor in latent space ($d = 2$)

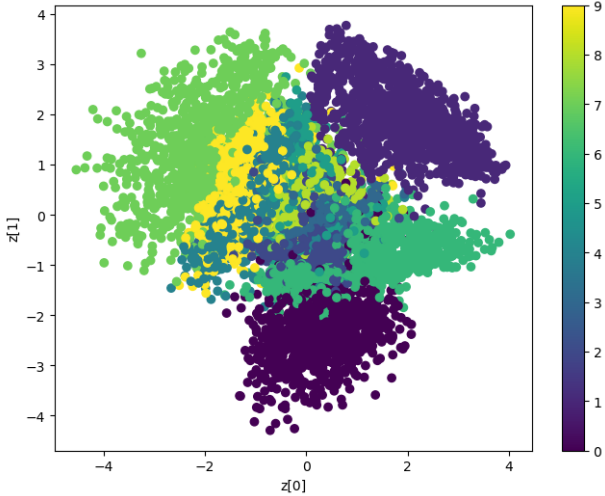


Figure: The test data is plotted in the latent space. Each point corresponds to an MNIST digit, and the color of the point indicates the digit's label.

Grid sampling from the normal distribution.
Recall that $q_\phi(\mathbf{z}|x) \approx p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$

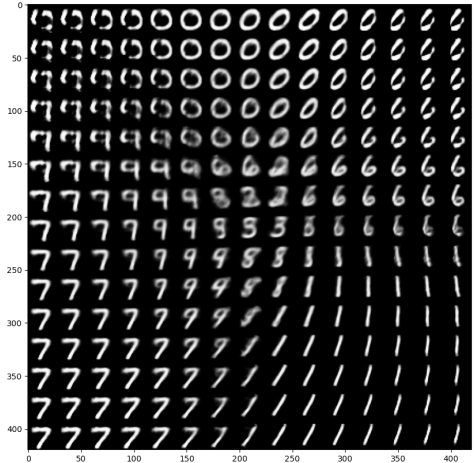
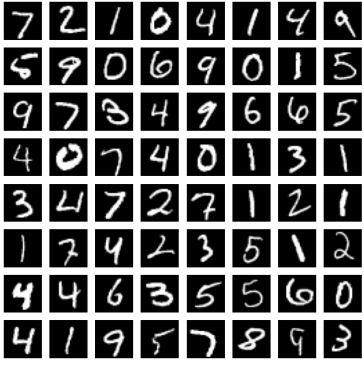
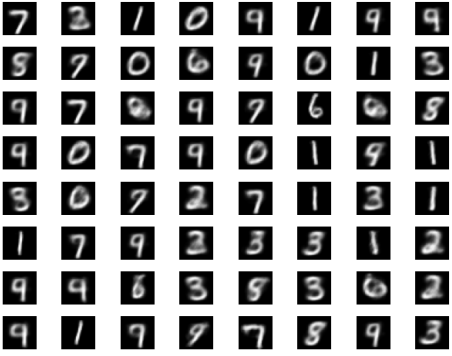


Figure: Each image is generated by sampling a point in the latent space and then using the generator (decoder) to produce an image.

Reconstruction results



(a) sample



(b) reconstruction = 128

Generations from random noises

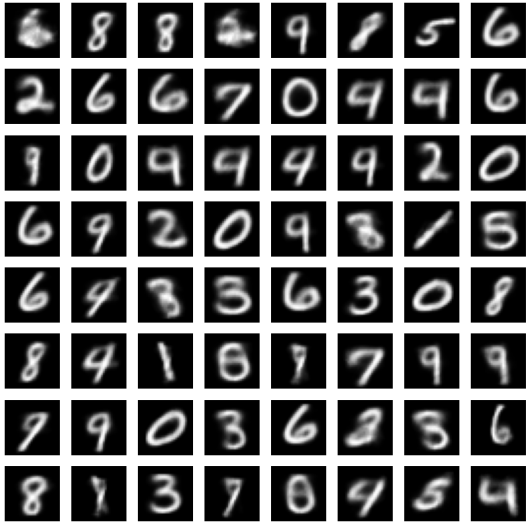


Figure: 64 samples from $\mathcal{N}(0, I)$

Try with CIFAR-10,

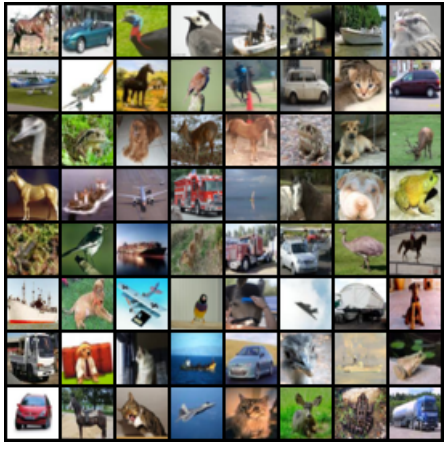


Figure: Randomly chosen 64 images from CIFAR-10

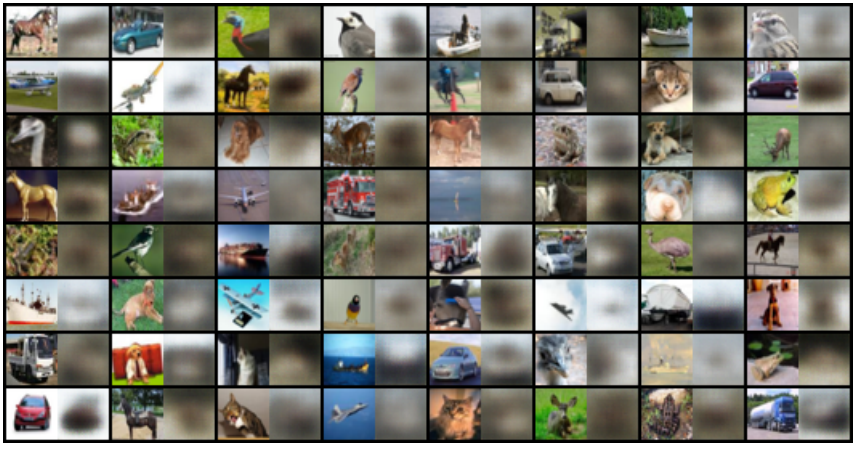


Figure: Images V.S. Reconstructions

FID Score

1. (FID) Given two distributions P and Q on R^d, we assume the they have finite first and second moments: $\mu_X = \int x dP < \infty, \mu_Y = \int y dQ < \infty$, and $\Sigma_X = \int xx^t dP < \infty, \Sigma_Y = \int yy^t dQ < \infty$.

$$FID(P,Q) = \inf \left\{ E|X-Y|^2; (X,Y) : X \sim P, Y \sim Q \right\}$$

This is the minimum expected square distance between two random variables in R^d with some joint distribution such that the marginal distribution of X is P and the marginal distribution of Y is Q.

Theorem: Assume that Σ_X, Σ_Y are non-singular, i.e. positive definite. Then

$$FID(P,Q) = |\mu_X - \mu_Y|^2 + \text{tr}\left[\Sigma_X + \Sigma_Y - 2\left(\Sigma_X^{1/2}\Sigma_Y\Sigma_X^{1/2}\right)^{1/2}\right]$$

Experiments

FID-Calculation The FID score we get is around 8.04

```
def fid_score(mu1, sigma1, mu2, sigma2):
    """Compute FID score given mean and covariance of two datasets."""
    diff = mu1 - mu2
    # covmean = sqrtm(sqrtm(sigma1) @ sigma2 @ sqrtm(sigma1))
    covmean = stable_covmean(sigma1, sigma2)
    # Real part is taken to avoid any potential imaginary number due to
    numerical issues
    fid = diff.dot(diff) + np.trace(sigma1 + sigma2 - 2 * covmean)
    return fid
```

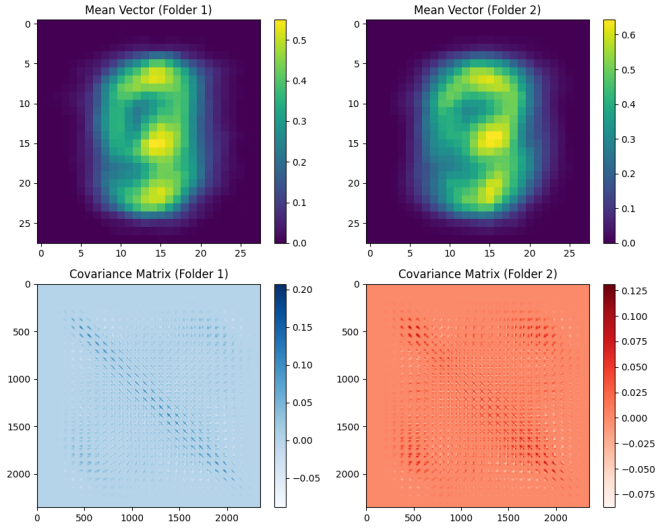


Figure: Computed means and Variances

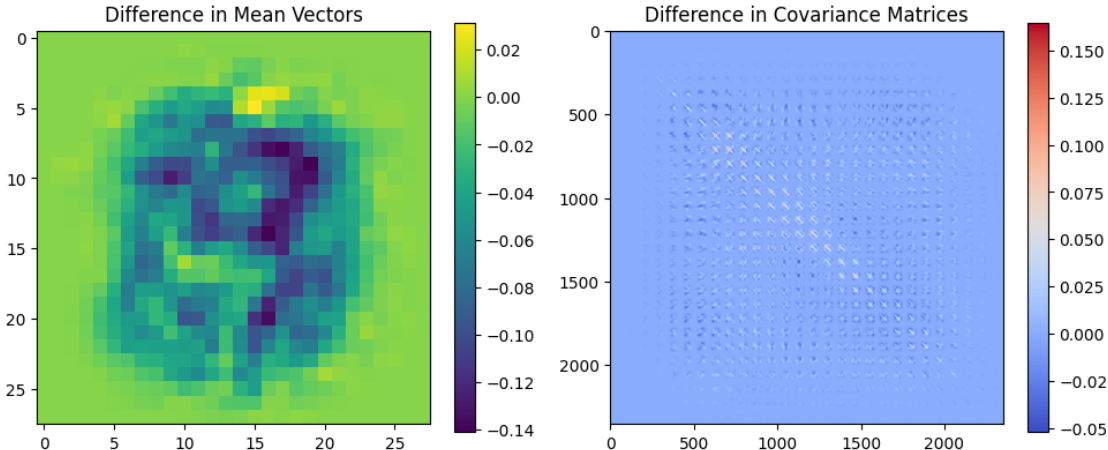


Figure: Differences in means and Variances

References

- Hinton, G. & Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science*. **313**, 504-507 (2006)
- Kingma, D. & Welling, M. Auto-encoding variational bayes. *ArXiv Preprint ArXiv:1312.6114*. (2013)
- David M. Blei, A. & McAuliffe, J. Variational Inference: A Review for Statisticians. *Journal Of The American Statistical Association*. **112**, 859-877 (2017),
- Weng, L. From Autoencoder to Beta-VAE. *Lilianweng.github.io*. (2018), https://lilianweng.github.io/posts/2018-08-12-vae/
- Variational Bayesian methods. Wikipedia. *https* : *//en.wikipedia.org/wiki/Variational_Bayesian_methods*
- Expectation–maximization algorithm. Wikipedia. *https* : *//en.wikipedia.org/wiki/Expectation*
- ethanluoyc, pytorch-vae, 2022, GitHub repository, https://github.com/ethanluoyc/pytorch-vae

End