Auto-Encoding Variational Bayes

CIDA Lab.

Presentation: Baekryun Seong

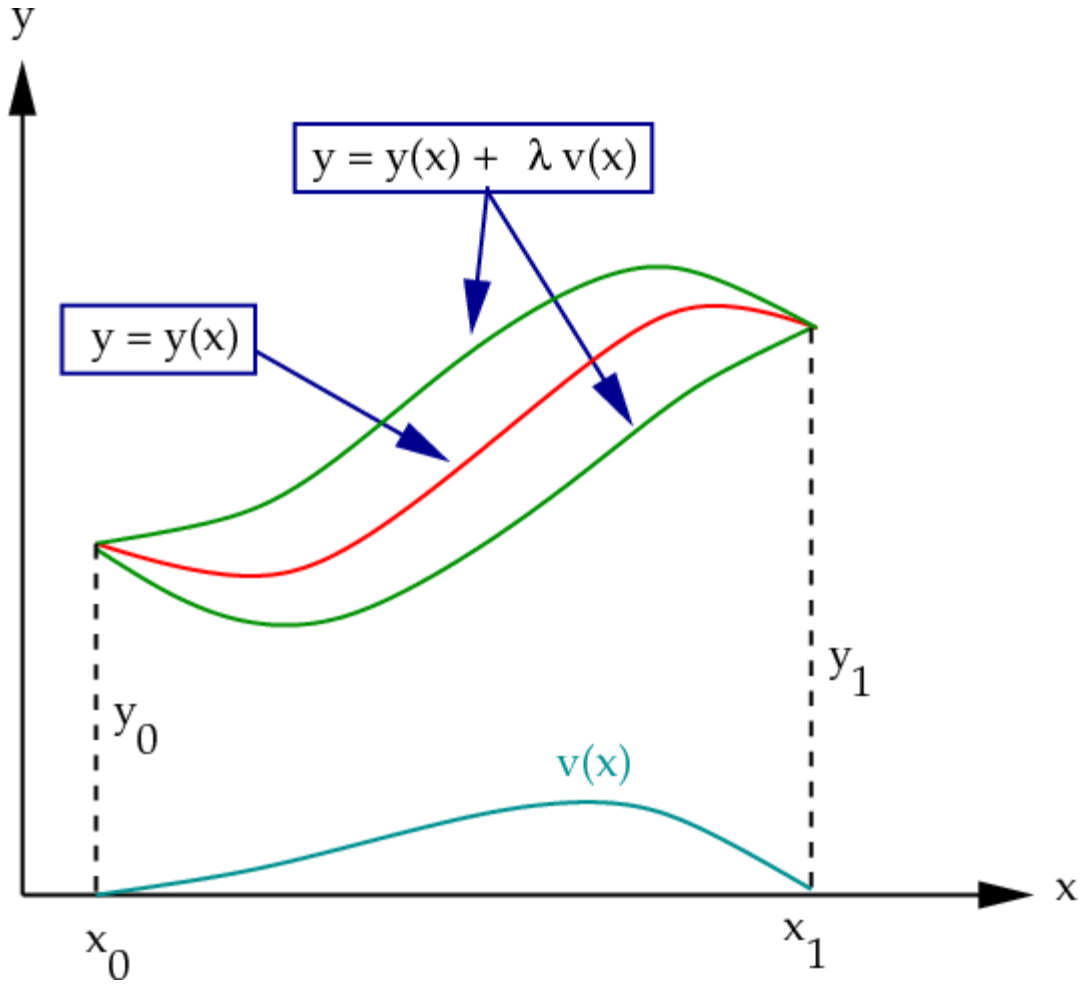
2024.07.05

Table of Contents

Intro: Diffusion Model

Diffusion model uses stacked Variational Auto Encoder.
But what is it? What is variational and what is auto encoder?

Calculus of Variations



https://en.wikiversity.org/wiki/Continuum_mechanics/Calculus_of_variations

Function is a mapping from domain to codomain.

$$f(x) = y$$

When domain is a set of functions, then it is called *functional*.

$$F[y(x)] = z$$

We already knows a well-known example of functional: Expectation.

$$E[X] = E[X(s)] = \int dx \cdot p(x) \cdot x$$

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

Calculus of Variations: Objective

In calculus, we generally find a x that maximizes y.

In variational calculus, we find $f(x)$ that maximizes functional F.

The derivative of functional F is:

$$\frac{\delta F}{\delta f(\boldsymbol{x})} = \lim_{\varepsilon \to 0} \frac{F[f(\boldsymbol{x}') + \varepsilon \delta(\boldsymbol{x}' - \boldsymbol{x})] - F[f(\boldsymbol{x}')]}{\varepsilon}$$

or it can be expressed other way:

$$\frac{\delta}{\delta f(\boldsymbol{x})} \int g(f(\boldsymbol{x}), \boldsymbol{x}) d\boldsymbol{x} = \frac{\partial}{\partial y} g(f(\boldsymbol{x}), \boldsymbol{x}) \quad \text{where } y = f(\boldsymbol{x}).$$

Treating function as vector $\boldsymbol{\theta}$ of infinite dimension,

$$\frac{\partial}{\partial \theta_i} \sum_j g(\theta_j, j) = \frac{\partial}{\partial \theta_i} g(\theta_i, i).$$

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
Lancaster, T., & Blundell, S. J. (2014). Quantum field theory for the gifted amateur. OUP Oxford.
Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

Calculus of Variations: Objective

Applying power series expansion,

$$F[f(x) + \epsilon\eta(x)] = F[f(x)] + \epsilon \int \frac{\delta F}{\delta f(x)} \eta(x) dx + O(\epsilon^2)$$

where ϵ is a small perturbation and η is a random function.

In critical point, F is stationary over any function η.

$$\int \frac{\delta F}{\delta y(x)} \eta(x) dx = 0$$

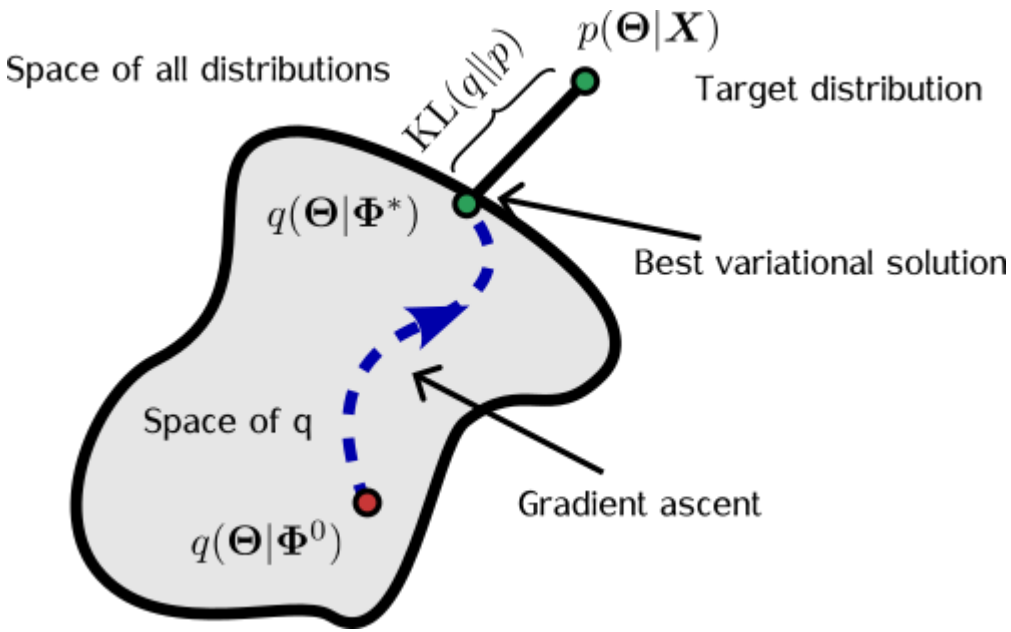
We can use these properties to prove some propositions. In example, we can prove that

The minimum-length curve between two points is straight line.

https://en.wikipedia.org/wiki/Calculus_of_variations

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

Variational Inference



https://towardsdatascience.com/variational-inference-the-basics-f70ac511bcea

We want to inference distribution of observable variables X and latent variables Z.

$$\log p(\boldsymbol{X}|\theta) = \mathcal{L}(q) + \mathrm{KL}(q||p)$$

$$\mathcal{L}(q) = \int q(\boldsymbol{Z}|\theta) \log \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\theta)}{q(\boldsymbol{Z}|\theta)} d\boldsymbol{Z}$$

$$\mathrm{KL}(q||p) = \int q(\boldsymbol{Z}|\theta) \log \frac{q(\boldsymbol{Z}|\theta)}{p(\boldsymbol{Z}|\boldsymbol{X}, \theta)} d\boldsymbol{Z} \geq 0 \quad \because \text{Gibbs inequality}$$

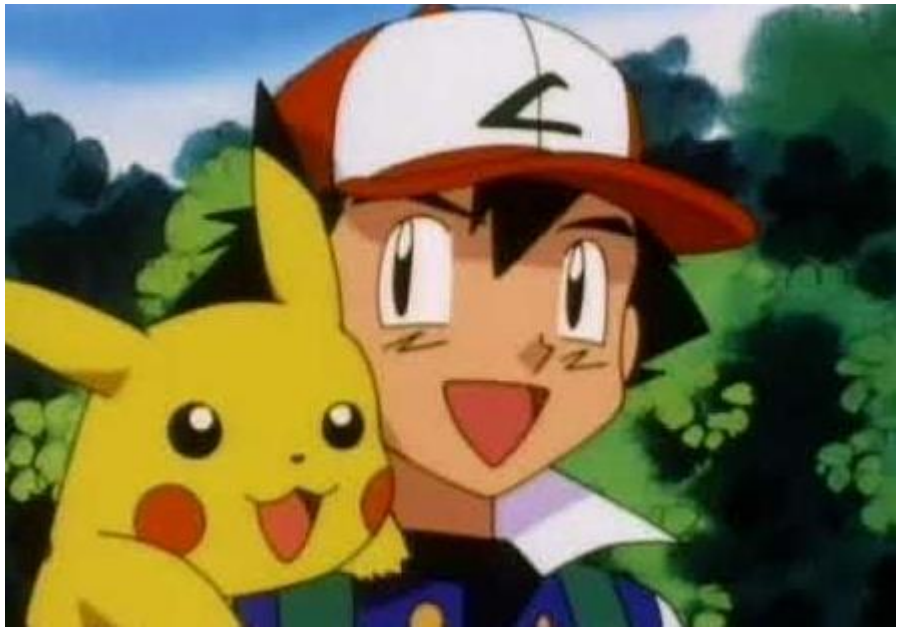
$p(\boldsymbol{X}|\theta)$ is usually called *evidence* of θ about $\boldsymbol{X}$.

Because of Gibbs inequality, $\log p(\boldsymbol{X}|\theta) \geq \mathcal{L}(q)$ is always satisfied. So $\mathcal{L}(q)$ is often written as *Evidence Lower BOund* (ELBO).

In k-means clustering, evidence $p(\boldsymbol{X}|\theta)$ monotonically improves by the property above.

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

Problem Scenario



Problem Scenario

$\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ is a dataset consists of N i.i.d. samples.

$\mathbf{X}$ is observable and $\mathbf{Z}$ is not.

Assume that the environment generates data sample by:

1. Generate $\mathbf{z}^{(i)}$ by prior distribution $p_{\theta^*}(\mathbf{z})$.
2. Generate $\mathbf{x}^{(i)}$ by conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z})$.

With assumptions above, the authors suggests a general solution works with:

1. The Bayesian integral is intractable, so the true posterior cannot be calculated. As a result, EM algorithm cannot be used. E.g. Kmeans.
2. Too large dataset to cluster using sampling-based solutions. (Monte Carlo EM)

Problem Scenario

Authors propose a solution to three related problems in the scenario:

1. Efficient approximate ML or MAP for the θ.
2. Efficient approximate posterior inference of latent variable $p_\theta(\mathbf{z}|\mathbf{x})$.
3. Efficient approximate marginal inference of the variable $\mathbf{x}$.

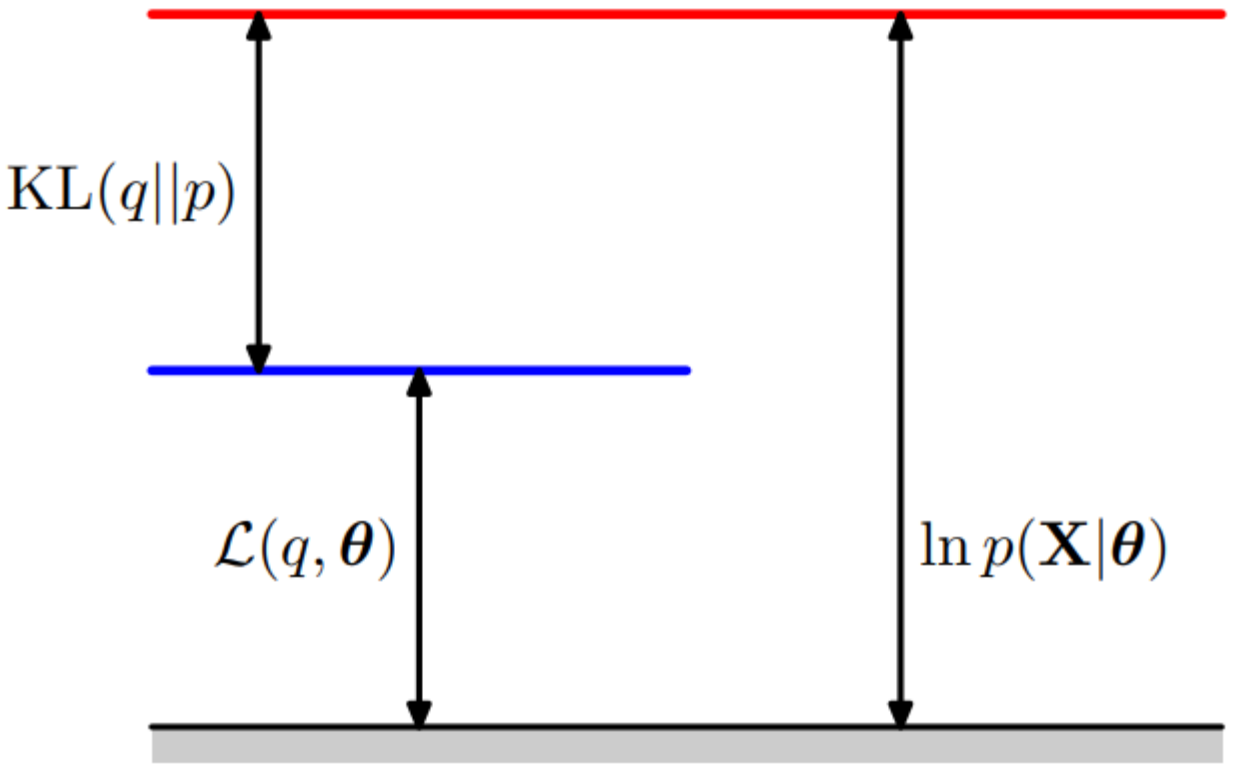
Define a recognition model $q_\phi(\mathbf{z}|\mathbf{x})$, an approximation of $p_\theta(\mathbf{z}|\mathbf{x})$, which is intractable.

Namely, it is *probabilistic encoder* that encodes $\mathbf{x}$ to $\mathbf{z}$.

In the perspective of coding theory, $\mathbf{z}$ is a code of $\mathbf{x}$.

In the same context, $p_\theta(\mathbf{x}|\mathbf{z})$ is a *probabilistic decoder*.

Variational Bound



Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

Variational Bound

We already saw variational inference that maximizes log likelihood:

$$\log p(\boldsymbol{X}|\theta) = \mathcal{L}(q) + \mathrm{KL}(q||p).$$

It is a kind of variational inference, because log likelihood is a functional that takes functions p, q as parameter.

Authors utilize it to approximate true posterior p_θ.

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

$\mathcal{L}(q)$ can be expressed with $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$ because L only depends on the two parameters.

Maximize the lower bound of log likelihood $\mathcal{L}$ only considering the function family: p_θ, q_ϕ.

Now because the distributions p_θ, q_ϕ only depends on θ, ϕ, the problem becomes general function optimizing problem, which can be solved by gradient descent.

Now we try to apply gradient descent:

$$\nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi}[f(\mathbf{z})] = \nabla_\phi \int d\mathbf{z} \cdot q_\phi\left(\mathbf{z}\middle|\mathbf{x}^{(i)}\right) f(\mathbf{z})$$

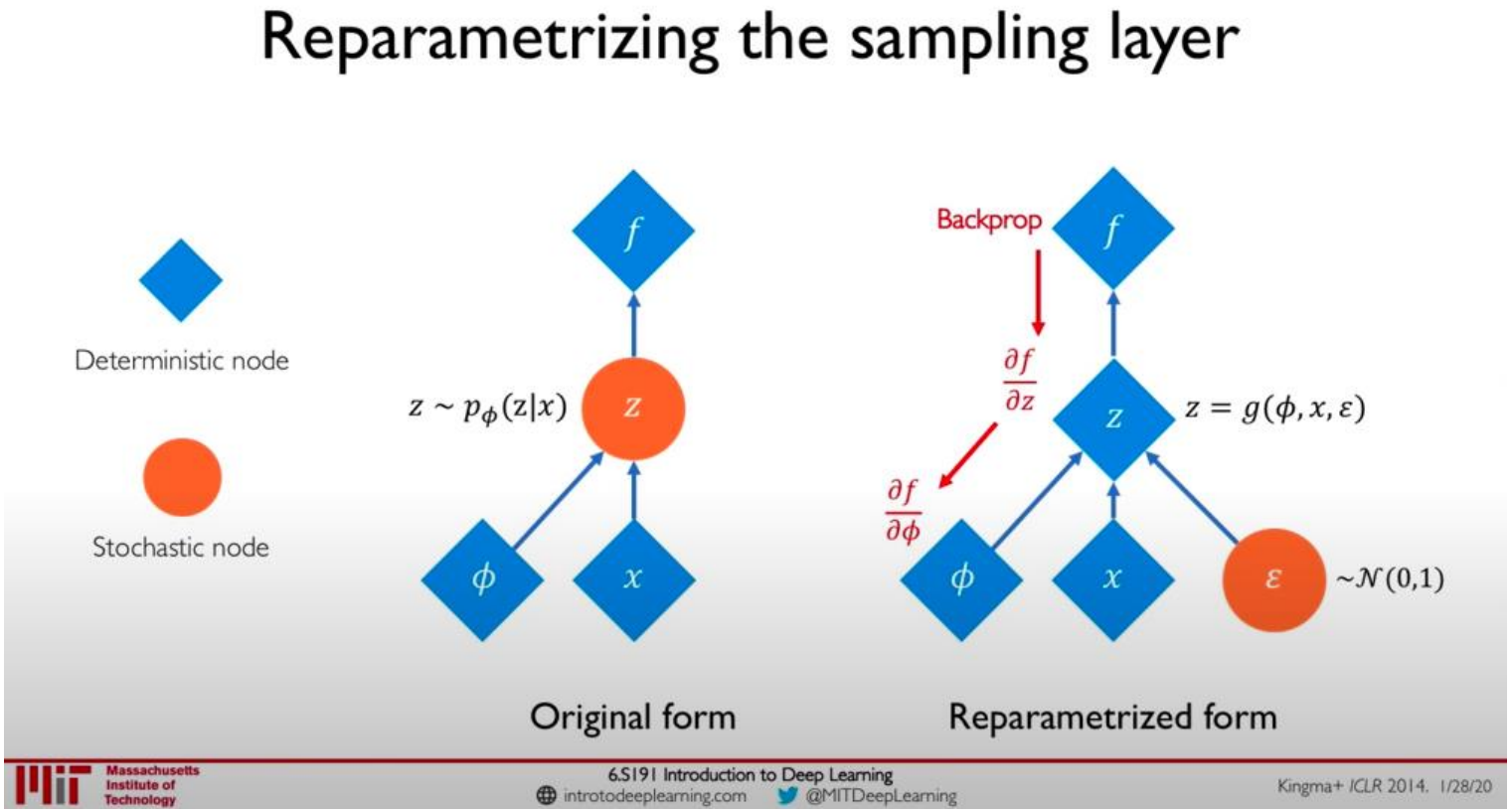
$$= \int d\mathbf{z} \cdot q_\phi\left(\mathbf{z}\middle|\mathbf{x}^{(i)}\right) \frac{\nabla_\phi q_\phi\left(\mathbf{z}\middle|\mathbf{x}^{(i)}\right)}{q_\phi\left(\mathbf{z}\middle|\mathbf{x}^{(i)}\right)} f(\mathbf{z})$$

$$= \int d\mathbf{z} \cdot q_\phi\left(\mathbf{z}\middle|\mathbf{x}^{(i)}\right) \cdot \nabla_\phi \log q_\phi\left(\mathbf{z}\middle|\mathbf{x}^{(i)}\right) f(\mathbf{z})$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi}\left[f(\mathbf{z}) \nabla_\phi \log q_\phi\left(\mathbf{z}\middle|\mathbf{x}^{(i)}\right)\right]$$

$$\simeq \frac{1}{L}\sum_{l=1}^{L} f\left(\mathbf{z}^{(l)}\right) \nabla_\phi \log q_\phi\left(\mathbf{z}^{(l)}|\mathbf{x}^{(i)}\right) \quad \because \text{Monte Carlo sampling.}$$

Reparameterization trick



Soleimany, A. "Deep Generative Modeling, MIT 6.S191." MIT Introduction to Deep Learning 6.S191: Lecture 4. 28 Feb 2020. https://www.youtube.com/watch?v=rZufA635dq4 Visited 26 May 2020.

Reparameterization trick

$$\frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z})\nabla_{\phi} \log q_{\phi}\left(\mathbf{z}^{(l)}\right)$$

But it is shown that gradient with Monte Carlo sampling has too high variance!* (**This problem also appears in REINFORCE, and other Monte Carlo based sampling methods**)

How can we reduce variance?

Idea: Instead of approximating q_{ϕ} itself, decompose it into deterministic and stochastic parts.

e.g. $\mathcal{N}(0, \sigma^2) = \sigma \cdot \mathcal{N}(0,1)$.

Now our target function is $g_{\phi}(\boldsymbol{\epsilon}, \mathbf{x})$, which is the deterministic part of $q_{\phi}(\mathbf{z}|\mathbf{x})$. And there is separated stochastic part of it: $p(\boldsymbol{\epsilon})$.

But what does it mean? 🤔

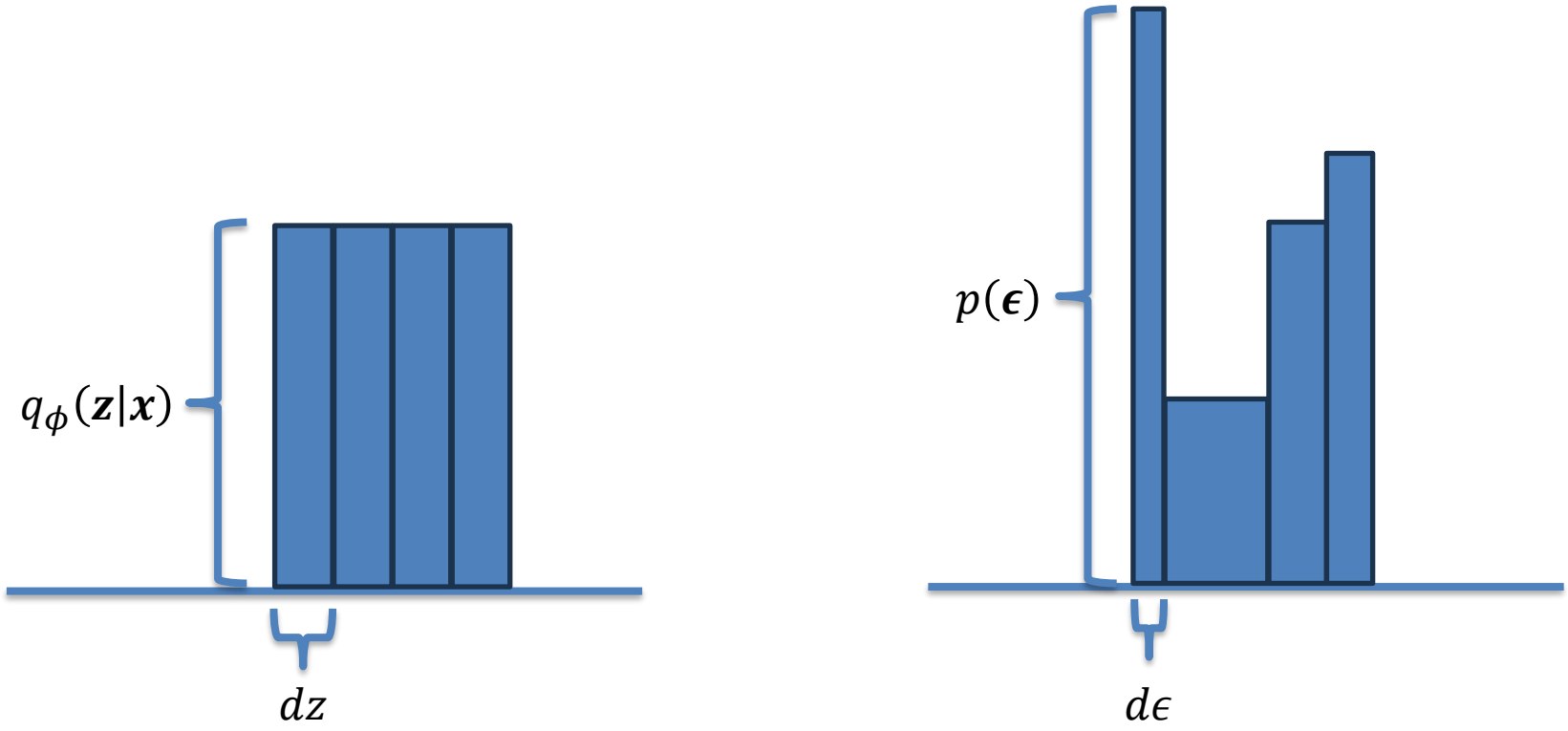
*Some blogs say that the why we use reparameterization is that we can not estimate gradient by Monte Carlo sampling, **but it is not true**.

Reparameterization trick

Given the deterministic mapping $\mathbf{z} = g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$, the expression below is satisfied:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) \prod_i dz_i = p(\boldsymbol{\epsilon}) \prod_i d\epsilon_i$$

Because the deterministic part has disappeared.



Reparameterization trick

So, $\int q_\phi(\mathbf{z}|\mathbf{x}) f(\mathbf{z}) d\mathbf{z} = \int p(\boldsymbol{\epsilon}) f(\mathbf{z}) d\boldsymbol{\epsilon} = \int p(\boldsymbol{\epsilon}) f\left(g_\phi(\boldsymbol{\epsilon}, \mathbf{x})\right) d\boldsymbol{\epsilon}$.

We can approximate the gradient of $\nabla_\phi g_\phi$ excluding the noise variable ϵ.

In other words, the variance is reduced:

$Var(SamplingNoise + LatentNoise)$ to $Var(SamplingNoise)$.

In example, we can reparametrize $\mathcal{N}(\mu, \sigma^2)$ to $\mu + \sigma\mathcal{N}(0,1)$.

Now, we can estimate μ and σ using the samples from $\mathcal{N}(0,1)$!

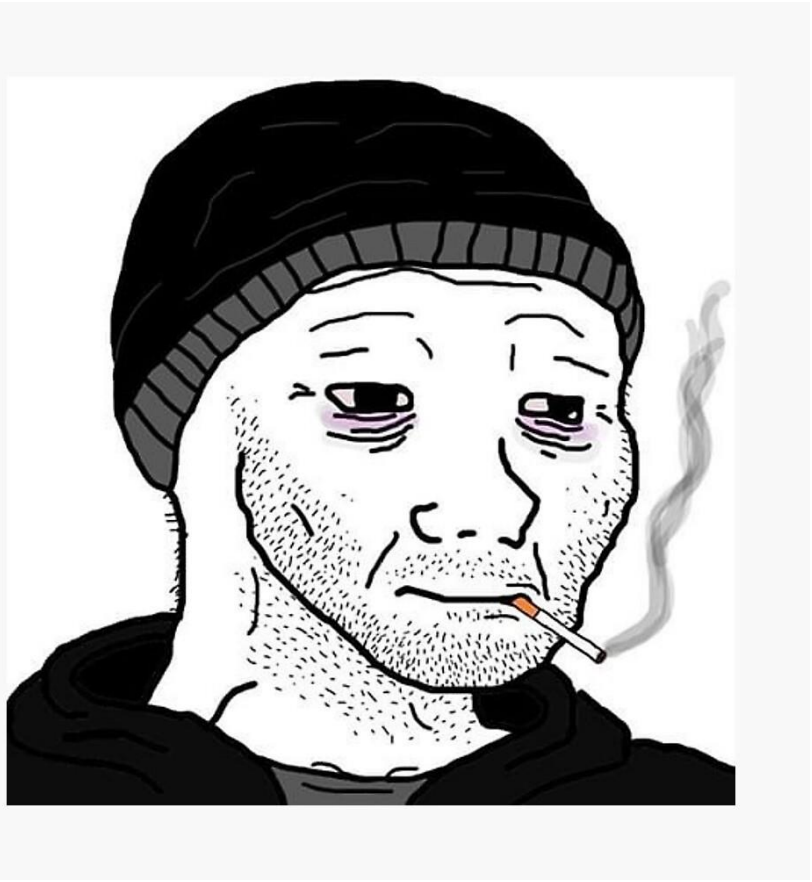
We must choose prior distribution of latent noise $p(\boldsymbol{\epsilon})$!

In general, it is normal distribution, because of central limit theorem.

> you cannot do inference without making assumptions.

MacKay, D. J. (2003). Information theory, inference and learning algorithms. Cambridge university press. p26

The SGVB estimator and AEVB algorithm



We can derive expression below

using $\int q_\phi(\mathbf{z}|\mathbf{x}) f(\mathbf{z}) d\mathbf{z} = \int p(\boldsymbol{\epsilon}) f\left(g_\phi(\boldsymbol{\epsilon}, \mathbf{x})\right) d\boldsymbol{\epsilon}$ and reparameterization trick.

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

In fact, we are trying to get:

$$\nabla_{\theta,\phi}\tilde{\mathcal{L}} = \nabla_{\theta,\phi}\frac{1}{L}\sum_{l=1}^{L}\left[\log p_\theta\left(x^{(i)}, g_\phi\left(\boldsymbol{\epsilon}^{(i,l)}, \boldsymbol{x}^{(i)}\right)\right) - \log q_\phi\left(g_\phi\left(\boldsymbol{\epsilon}^{(i,l)}, \boldsymbol{x}^{(i)}\right)\middle| x^{(i)}\right)\right].$$

Where the original ELBO is:

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

$\nabla_{\theta,\phi}\tilde{\mathcal{L}}$ is stochastic gradient variational bayes estimator.

Auto Encoding Variational Bayes Algorithm

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 $\boldsymbol{\theta}, \boldsymbol{\phi}$

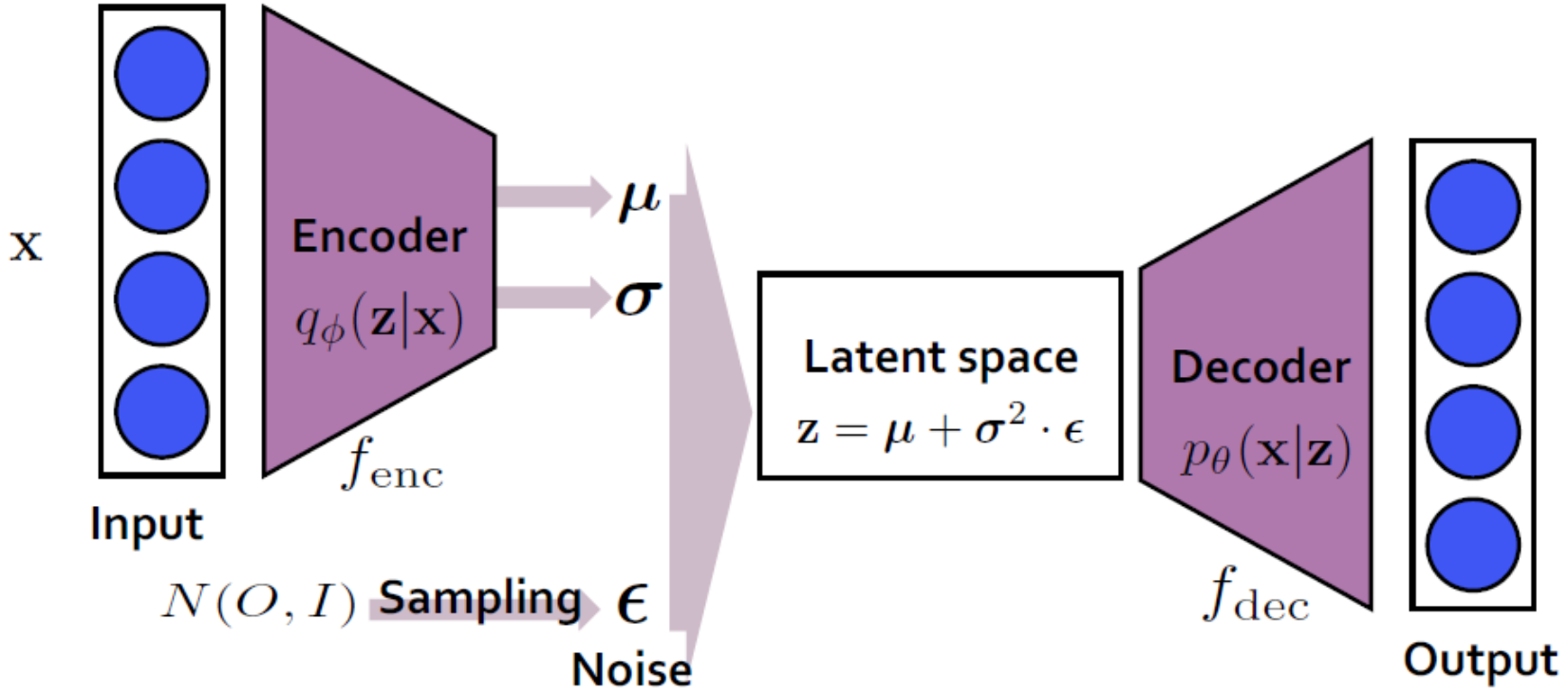
Sometimes, $\tilde{\mathcal{L}}$ can be integrated exactly. Then SGVB becomes:

$$\tilde{\mathcal{L}} = -\mathrm{KL}(q_\phi || p_\theta) + \frac{1}{L}\sum_{l=1}^{L}\left[\log p_\theta\left(x^{(i)}, g_\phi\left(\boldsymbol{\epsilon}^{(i,l)}, \boldsymbol{x}^{(i)}\right)\right)\right].$$

If p_θ, q_ϕ are both Gaussian,

$$\tilde{\mathcal{L}} = -\frac{1}{2}\sum_{j=1}^{J}\left[1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2\right] + \frac{1}{L}\sum_{l=1}^{L}\left[\log p_\theta\left(x^{(i)}, g_\phi\left(\boldsymbol{\epsilon}^{(i,l)}, \boldsymbol{x}^{(i)}\right)\right)\right].$$

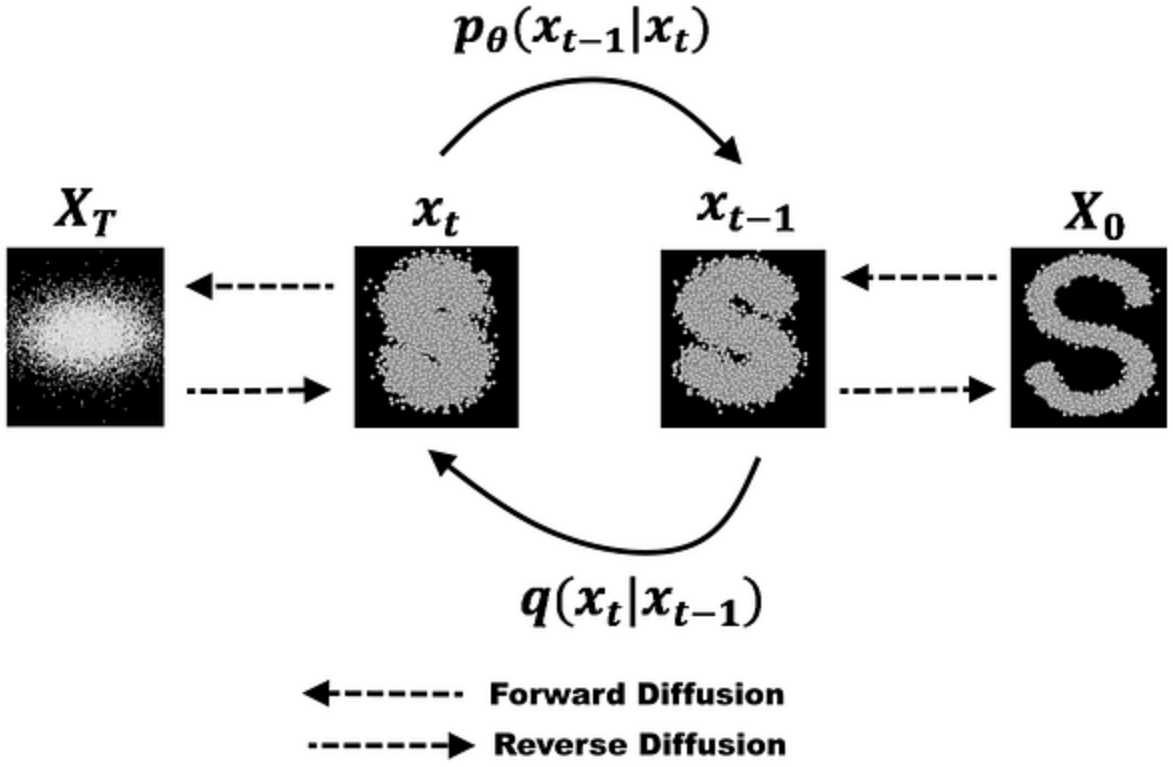
Variational Auto Encoder



Code:

[minimal_vae/vae_mnist.py at master · adler-j/minimal_vae · GitHub]

Outro: Diffusion Model



https://towardsdatascience.com/diffusion-models-made-easy-8414298ce4da

Shortcomings

Generated samples from VAE are blurry, in general.

Maybe the reason of it is the limitation of maximizing log likelihood.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.