14. Variational Autoencoder

"Deep Learning" Course (2020)

Takyoung Kim

This is an informal lecture review of Prof. Sungroh Yoon @ Seoul National University.
I would like to express my appreciation to the professor for providing an insightful lecture.
Lectures are publicly opened in this YouTube.

Table of Contents

List of Tables

List of Figures

1. Autoencoders

1.1 Autoencoder (AE)

An autoencoder is a neural network trained to reconstruct input x as output $\tilde{x}$ (i.e., $x \sim \tilde{x}$) in unsupervised way. That is, what we want to learn is "code", an efficient representation of input data. Table 1 indicates an architecture of the autoencoder.

Table 1. Autoencoder architecture. Variational autoencoder will use stochastic encoder/decoder.

	deterministic	**stochastic**	**parametrized by**
encoder	$f_\phi(x) = h$	$q_{\text{enc}}(h\|x)$	ϕ
decoder	$g_\theta(h) = \tilde{x}$	$p_{\text{dec}}(\tilde{x}\|h)$	θ

A dimension of code is smaller than the original data (i.e., $dim(x) > dim(h)$), so that the code can capture important features of input. Examples of its applications are following:

- dimensionality reduction
- feature learning
- forefront of generative modeling

While training, parameters ϕ^* and θ^* are optimized by loss, such as mean squared error. In the case of linear encoder/decoder, the subspace of code h spans the same subspace as principal component analysis (PCA). Deep AE or stacked AE can be constructed using multiple hidden layers.

1.2 Need for regularization

However, encoder and decoder with too much capacity just copy input to output. It learns nothing useful about data distribution. Therefore we limit model capacity for regularizing and use a loss encouraging desirable properties. For example, denoising autoencoder (DAE) intends to make model robust to noise or missing inputs. Sparse autoencoder (SAE) makes sparse representation (code), and contractive autoencoder maps similar input into similar representation.

1.3 Denoising autoencoder (DAE)

DAE is one way to force AE to learn useful features. It adds noise to inputs, then trains AE to recover the original (noise-free) inputs. Gaussian noise or dropout are some examples of noise.

2. Variational autoencoder: Introduction

2.1 Deep generative learning

Deep generative learning can be categorized into two parts: likelihood-based models and implicit models. Examples of each category are written in the following:

1. likelihood-based frameworks
    - autoregressive models
    - variational autoencoders
    - flow-based models
    - diffusion models
2. implicit models
    - generative adversarial networks

2.2 Variational autoencoder (VAE)

VAE is a popular approach to unsupervised learning of complex distributions. It uses learned approximate inference, and can be trained purely with gradient-based methods (e.g., SGD). Its easy use appeals many researchers because it can be built on top of standard function approximators (i.e., neural nets).

Comparing with other likelihood-based models, VAE has advantages of fast/tractable sampling and easy-to-access encoding networks. However, its sampling quality may not be the best.

Table 2. Comparing autoregressive model, VAE, and GAN.

model	**modeling $p(x)$**	**attributes**
Autoregressive	$p(x) = \sum_{i=1}^{n} p(x_i \mid x_1, \dots, x_{i-1})$	high quality but slow
VAE	$p(x) = \int p(z)p(x \mid z)dz$	assuming latent variable z
GAN	give up explicit $p(x)$ modeling	just sample from it

2.3 Related models

Autoencoder reconstructs given input x when training, and then the encoder is used as feature learner for downstream learning.

Generative models generate new sample x from a certain distribution. Its main challenge is that it is hard to model dependencies between dimensions. The left figure in Figure 1 shows the situation.

Latent generative models generate new sample x from latent z to ease this difficulty. Introducing z improves dependency modeling, and it can be trained by maximizing data likelihood $p(x) = \int p(z)p(x|z)dz$. However, it also has a limitation that the integration $\int dz$ is often intractable.

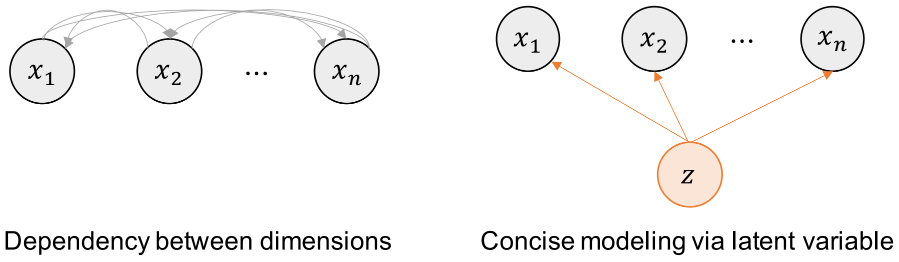

x_1
x_2
...
x_n
Dependency between dimensions
x_1
x_2
...
x_n
z
Concise modeling via latent variable

Fig. 1. Modeling via latent variable

2.4 Desideratum in latent space

A requirement for standard AE is **seperation** and **clustering**. It should be grouped into same classes, so that features can be utilized in an appropriate way. However, a requirement for VAE is **continuity** and **completeness**, because it should sample from latent z to generate output. Figure 2 shows that VAE's latent space tends to be spread compared to AE's.

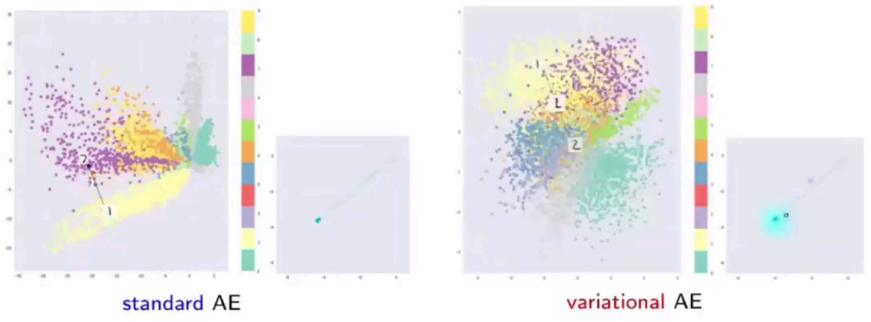


Fig. 2. Latent space of AE and VAE

2.5 Idea behind VAE

Figure 3 shows the overall training process of VAE. We want to map input data to a simple distribution (e.g., Gaussian) as a form of distribution. Then sampled latent variable is used

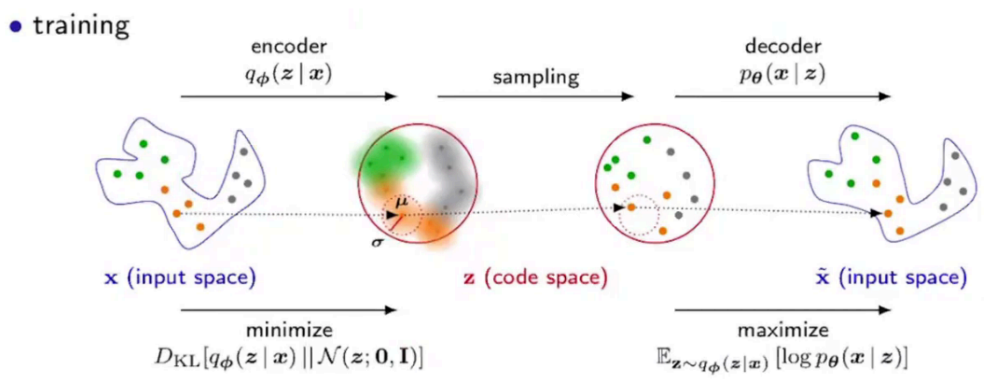

• training
encoder
$q_\phi(z \mid x)$
sampling
decoder
$p_\theta(x \mid z)$
μ
σ
x (input space)
z (code space)
$\tilde{x}$ (input space)
minimize
$D_{KL}[q_\phi(z \mid x) \,||\, \mathcal{N}(z; 0, I)]$
maximize
$\mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x \mid z)]$

Fig. 3. VAE: training phase

to reconstruct the original data. VAE's decoder tries to maximize likelihood of reconstructing data, while encoder regularizes distribution as a simple one. That is, a decoder is main focus of VAE, while AE focuses on its encoder.

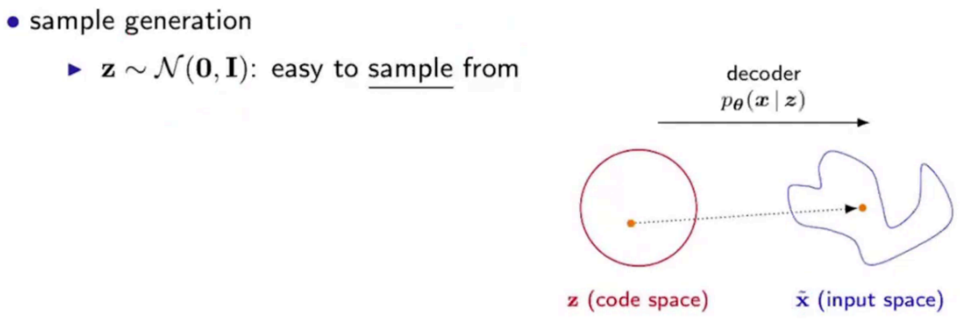

• sample generation
▸ $z \sim \mathcal{N}(0, I)$: easy to sample from
decoder
$p_\theta(x \mid z)$
z (code space)
$\tilde{x}$ (input space)

Fig. 4. VAE: inference phase

In inference phase, well-trained decoder would generate a nice output. Therefore, we choose any sample from code space, which is distribution easy to sample.

2.6 Preview of VAE formulation

In previous 'Introduction to generative models' chapter, we discussed about variational inference, which turns an inference problem into an optimization. VI can be formulated as follows:

$$\log p(x) - D_{KL}[q(z) \parallel p(z|x)] = \mathbb{E}_z[\log p(x|z)] - D_{KL}[q(z) \parallel p(z)] \tag{1}$$

The left terms are intractable, so we should optimize by maximizing the tractable right terms (ELBO).

Meanwhile, VAE can be formulated as *amortized* variational inference[1]:

$$\log p(x) - D_{KL}\left[q_\phi(z|x) \parallel p(z|x)\right] = \mathbb{E}_z\left[\log p_\theta(x|z)\right] - D_{KL}\left[q_\phi(z|x) \parallel p(z)\right] \tag{2}$$

It is the same in that VAE maximizes ELBO, but it utilizes the autoencoder architecture. ϕ and θ means variational parameters and model parameters, respectively. Maximizing ELBO w.r.t. ϕ and θ will concurrently optimize the two:

1. approximately maximize $p_\theta(x)$: our generative model becomes better
2. minimize KLD of approximation $q_\phi(z|x)$ from true posterior $p(z|x)$
   : our inference model $q_\phi(z|x)$ becomes better

3. Variational autoencoder: Formulation and architecture

3.1 Problem formulation

The objective of VAE is to maximize the probability of each x in training set under the entire generative process.

$$p(x) = \int p(z)p(x|z)dz \tag{3}$$

To solve above equation, VAEs must deal with three problems:

1. how to define latent variable z (i.e., decide what information they represent)
2. how to define output distribution $p(x|z)$
3. how to deal with integral over z

3.2 Solution 1: prior $p(z)$

We choose prior $p(z)$ to be simple because of its easy engineering attribute. That is, a simple distribution is reasonable to deal with disentangled latent attributes (e.g., pose, how much smile).

A common choice is the unit Gaussian $p(z) = \mathcal{N}(z; 0, \mathbf{I})$. Because of its diagonal prior attribute, there exist independent latent variables. These variables of different dimensions play a role of interpretable factors of variation.

3.3 Solution 2: output distribution $p(x|z)$

To make output distribution, a differentiable neural network $g(z;\theta)$ as a decoder is used. θ denotes parameters of decoder neural network, and $g(z;\theta)$ produces parameters of $p(x|z)$. Note that decoder output is parameters of distribution, not an output sample itself.

[1]Amortized inference is relatively more effieicnt to large data because it uses one set of parameters to model the relation between x and z, while standard VI models each data sample with its corresponding z

$$p(x|z) = p(x; g(z;\theta)) \triangleq p_\theta(x|z) \tag{4}$$

Common parametric distributions for $p(x|z)$ can be follows:

- continuous x: a Gaussian parametrized by $g(z;\theta)$
- binary x: a Bernoulli parametrized by $g(z;\theta)$

At training time, a likelihood of each input $x^{(i)}$ can be computed using $p_\theta(x^{(i)}|z)$, and its error is optimized by backpropagation while performing SGD. After training, a new x is **sampled** from $p_\theta(x|z)$. Figure 5 shows the overall process.

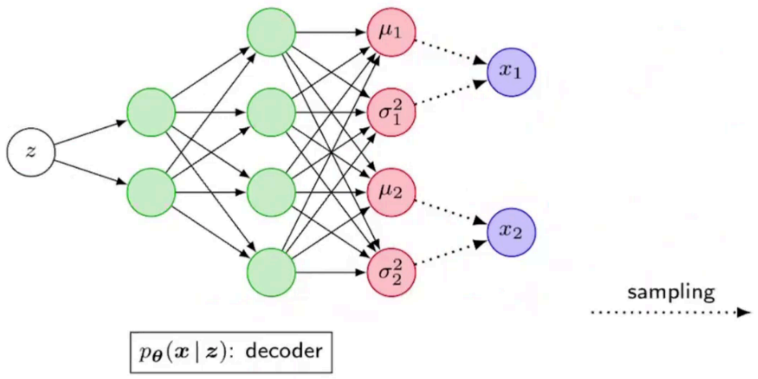


Fig. 5. VAE decoder

3.4 Solution 3: integration over z

From previous solutions, we discussed about what to choose prior $p(z)$ and how to optimize decoder $p_\theta(x|z)$.

$$p_\theta(x) = \int p(z)p_\theta(x|z)dz \tag{5}$$

However, the remaining problem is that integration over z is **intractable**. Conceptually, we can compute $p(x)$ approximately by sampling method. Like below, approximated $p(x)$ can be calculated by taking mean of sample outputs.

$$p(x) \approx \frac{1}{n}\sum_{i=1}^{n} p(x|z_i) \tag{6}$$

The main challenge of this approach is that n should be extremely large to have an accurate estimate of $p(x)$.

In practice, the volume data takes in high-dimensional space is very little. In other words, for most z, it contributes nothing to estimate of $p(x)$ (i.e., $p(x|z) \approx 0$).

3.4.1 Key idea behind VAE 1: using $p(z|x)$

An informative samples of z are likely to have produced x. So we can take a value of x, then get a distribution over z that are **likely to produce** x. However, it is impossible to use posterior $p(z|x) = p(x|z)p(z)/p(x)$ because of its intractability. Therefore, we need a new function $q(z|x)$.

3.4.2 Key idea behind VAE 2: variational inference

We can find an approximate posterior $q(z|x)$ using **variational inference**. That is, we try to maximize ELBO(q) rather than maximizing $p(x)$ directly.

3.4.3 Key idea behind VAE 3: amortized approach

Traditional variational inference is slow and requires closed form. It assumes variational distribution over every input data, which causes lack of scalability. To ease this assumption, an **amortized** variational inference is proposed.

Intuitively speaking, it is just using an **encoder** network with parameter ϕ. An amortized variational inference means that it takes every data with a single set of ϕ. The encoder $f(x;\phi)$ produces parameters of $q(z|x)$, formulated as follows:

$$q(z|x) = q(z; f(x;\phi)) \triangleq q_\phi(z|x) \tag{7}$$

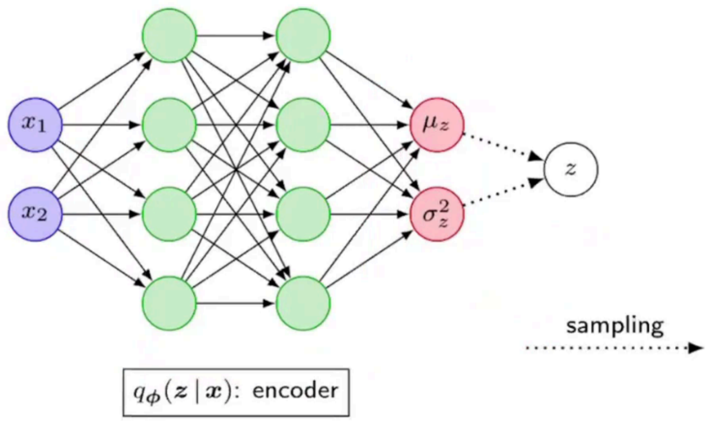


Fig. 6. VAE encoder

Training the encoder is easier than optimization in traditional variational inference, and can use SGD with backpropagation. A Gaussian distribution parametrized by $f(x;\phi)$ is commonly chosen for for $q(z|x)$. For computational issues, its covariance is constrained to be diagonal.

In training, z is sampled from $q_\phi(z|x)$, while sampled from $\mathcal{N}(0,\mathbf{I})$ after training. The sampled z is fed into decoder to generate x. Figure 6 illustrates the encoder architecture.

3.5 VAE architecture

The encoder and decoder are stacked, constructing VAE architecture in Figure 7. We can learn parameters ϕ and θ via backpropagation. Note that both encoder and decoder generate parameters.

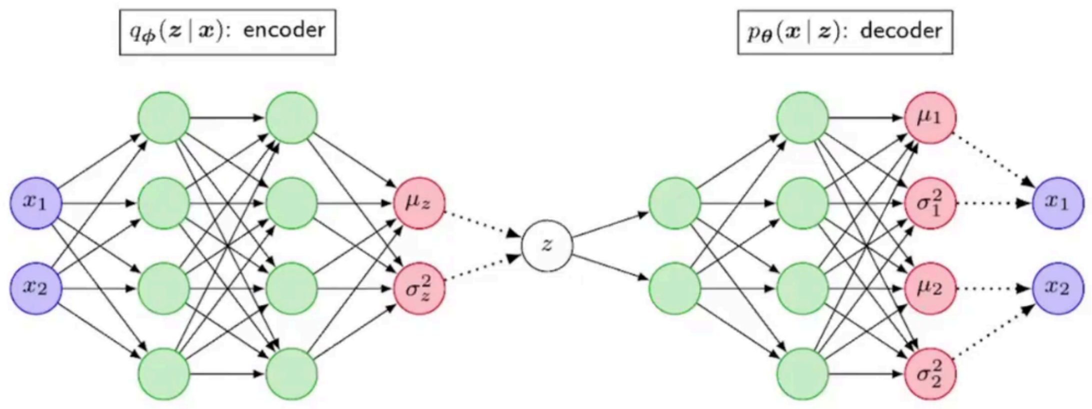
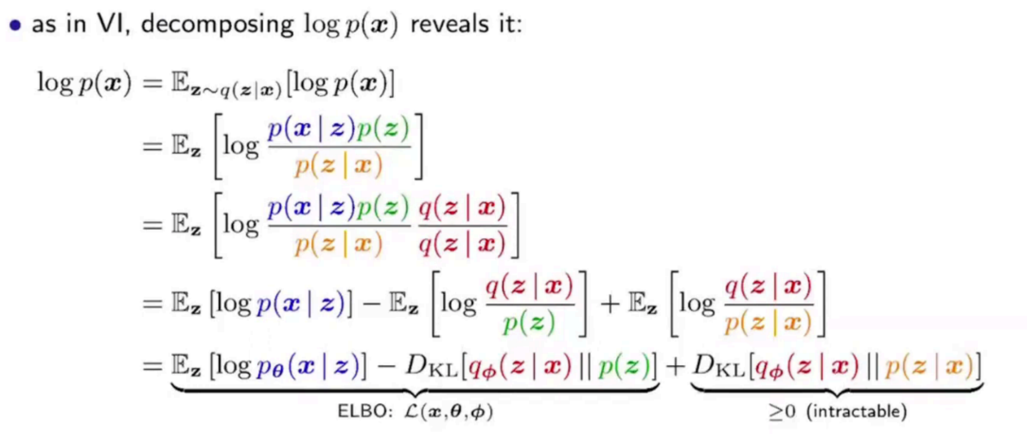


Fig. 7. The overall VAE architecture

4. Variational autoencoder: Training

4.1 Training objective

As in variational inference, we can decompose $\log p(x)$ according to Figure 8.



- as in VI, decomposing $\log p(\boldsymbol{x})$ reveals it:

$$\begin{aligned}
\log p(\boldsymbol{x}) &= \mathbb{E}_{\mathbf{z}\sim q(\boldsymbol{z}\mid\boldsymbol{x})}[\log p(\boldsymbol{x})] \\
&= \mathbb{E}_{\mathbf{z}}\left[\log \frac{p(\boldsymbol{x}\mid\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{z}\mid\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\mathbf{z}}\left[\log \frac{p(\boldsymbol{x}\mid\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{z}\mid\boldsymbol{x})}\frac{q(\boldsymbol{z}\mid\boldsymbol{x})}{q(\boldsymbol{z}\mid\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\mathbf{z}}[\log p(\boldsymbol{x}\mid\boldsymbol{z})] - \mathbb{E}_{\mathbf{z}}\left[\log \frac{q(\boldsymbol{z}\mid\boldsymbol{x})}{p(\boldsymbol{z})}\right] + \mathbb{E}_{\mathbf{z}}\left[\log \frac{q(\boldsymbol{z}\mid\boldsymbol{x})}{p(\boldsymbol{z}\mid\boldsymbol{x})}\right] \\
&= \underbrace{\mathbb{E}_{\mathbf{z}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}\mid\boldsymbol{z})] - D_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{z}\mid\boldsymbol{x})\,||\,p(\boldsymbol{z})]}_{\text{ELBO: } \mathcal{L}(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\phi})} + \underbrace{D_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{z}\mid\boldsymbol{x})\,||\,p(\boldsymbol{z}\mid\boldsymbol{x})]}_{\geq 0 \text{ (intractable)}}
\end{aligned}$$

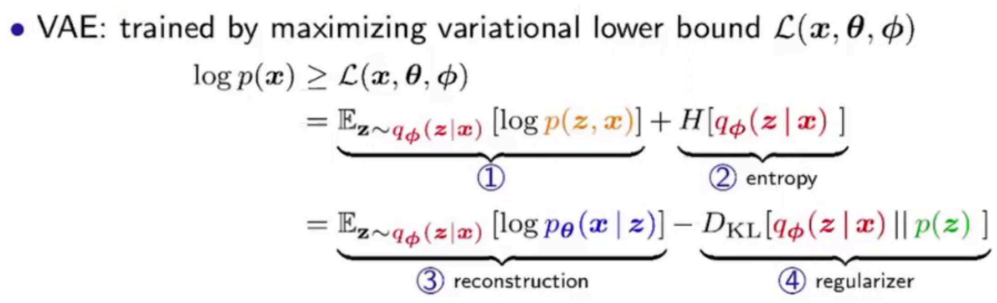
Fig. 8. VAE optimization

Different from traditional VI, VAE uses $q(z|x)$ instead of $q(z)$. It means that encoder is now a (stochastic) function of input x.

4.2 Interpretation

There are some perspectives interpreting VAE.

4.2.1 Perspective 1



- VAE: trained by maximizing variational lower bound $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi})$

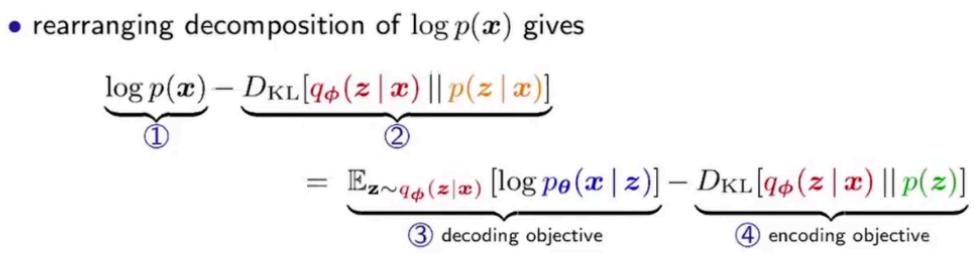
$$\begin{aligned}\log p(\boldsymbol{x}) &\geq \mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi}) \\ &= \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})}\left[\log p(\boldsymbol{z}, \boldsymbol{x})\right]}_{①} + \underbrace{H[q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})\,]}_{② \text{ entropy}} \\ &= \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})\right]}_{③ \text{ reconstruction}} - \underbrace{D_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z})\,]}_{④ \text{ regularizer}}\end{aligned}$$

Fig. 9. ELBO interpretation

On one hand, Figure 9 analyzes ELBO as reconstruction/regularizer perspective. Each term can be interpreted as follows:

1. joint log-likelihood of visible/hidden variables (under approximate posterior over latent variables)
2. entropy of approximate posterior: maximizing this encourages to
   - place high prob mass on many z values that could have generated x
   - rather than collapsing to a single point estimate of the most likely value
3. reconstruction log-likelihood found in other autoencoders
4. tries to make approximate posterior $q_\phi(z|x)$ and model prior $p(z)$ close
   - tries to reflect prior knowledge (as a regularizer)

4.2.2 Perspective 2



- rearranging decomposition of $\log p(\boldsymbol{x})$ gives

$$\begin{aligned}&\underbrace{\log p(\boldsymbol{x})}_{①} - \underbrace{D_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z} \mid \boldsymbol{x})]}_{②} \\ &\quad = \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})\right]}_{③ \text{ decoding objective}} - \underbrace{D_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z})]}_{④ \text{ encoding objective}}\end{aligned}$$

Fig. 10. Another ELBO interpretation

On the other hand, Figure 10 interprets VAE as encoder/decoder objectives.

1. (left) quantity we want to maximize
2. (left) error term (this will become small if q is high-capacity) $\rightarrow$ makes q produce z's that can reproduce a given x
3. (right) q is "encoding" x into z
4. (right) p is "decoding" it to reconstruct x

The right hand side is something we can optimize via SGD (given right choice of q.

4.3 Training

We can now train VAE using SGD to maximize the objective.

$$J = L(x,\theta,\phi) = -D_{KL}\left[q_\phi(z|x) \parallel p(z)\right] + \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \tag{8}$$

Usually we choose $q(z|x)$ as Gaussian, which means $q(z|x) = \mathcal{N}(z;\mu(x),\Sigma(x))$. μ and Σ are learned and represented via neural networks, and Σ is a diagonal matrix. The first encoding objective including D_{KL} between two multivariate Gaussians can be computed in closed form. However, the second decoding objective involves random sampling, making backpropagation impossible.

4.4 Training: encoding objective

The encoding objective, $-D_{KL}\left[q_\phi(z|x) \parallel p(z)\right]$, can be calculated in a closed form. Because we assumed diagonal form of covariance matrix, the objective can be simplified as below:

$$D_{KL}\left[\mathcal{N}(\mu(x)\Sigma(x)) \parallel \mathcal{N}(0,\mathbb{I})\right] \tag{9}$$

$$= \frac{1}{2}\left(tr(\Sigma(x)) + \mu(x)^T\mu(x) - K - \ln\det\Sigma(x)\right) \tag{10}$$

$$= \frac{1}{2}\left(\sum_{k=1}^{K}\sigma_k^2(x) + \sum_{k=1}^{K}\mu_k^2(x) - \sum_{k=1}^{K}1 - \ln\prod_{k=1}^{K}\sigma_k^2(x)\right) \tag{11}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\left(\sigma_k^2(x;\phi) + \mu_k^2(x;\phi) - 1 - \ln\sigma_k^2(x;\phi)\right) \tag{12}$$

It is differentiable and can be minized w.r.t. ϕ vis SGD.

4.5 Training: decoding objective

To compute decoding objective, $\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$, we use Monte Carlo sampling for each input $x^{(i)}$. First, $z^{(i,l)}$ is sampled from encoder $q_\phi(z|x^{(i)})$, and then fed to decoder. It provides $\log p_\theta(x^{(i)}|z^{(i,l)})$, repeating L times to approximate. It can be formulated as follows (in practice, L is often set to 1):

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x^{(i)}|z^{(i,l)})\right] \approx \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(x^{(i)}|z^{(i,l)}) \tag{13}$$

However, maximizing the above objective using SGD is impossible because it relies on sampling, which is neither continuous nor differentiable. So we need a trick called **reparameterization**.

4.6 Reparameterization trick

Reparameterization trick is a method to enable optimizing VAE architecture by moving sampling procedure to an input layer. Instead of sampling $z^{(i,l)} \sim \mathcal{N}(\mu(x^{(i)}), \Sigma(x^{(i)}))$ directly, we can compute $z^{(i,l)}$ by:

1. first sampling $\varepsilon^{(l)} \sim \mathcal{N}(0, \mathbb{I})$
2. then computing
$$z^{(i,l)} = \mu(x^{(i)}) + \sigma(x^{(i)}) \odot \varepsilon^{(l)} \tag{14}$$

Through this process, we can get $z^{(i,l)}$ with the same distribution as before, but now we can do backprop. Figure 11 illustrates the trick.

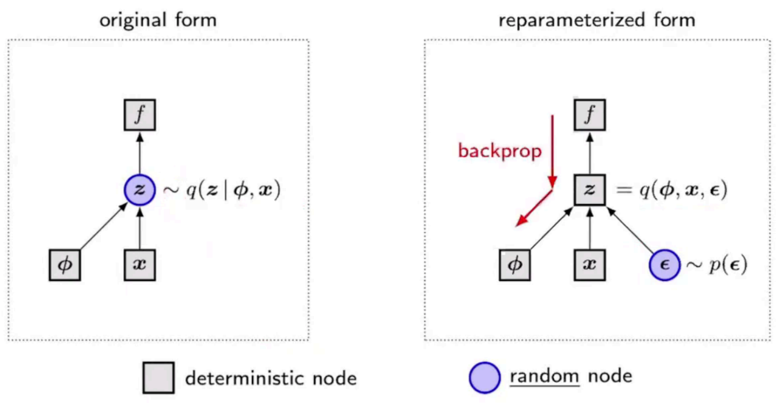


Fig. 11. Reparametrization trick

4.7 Training: combining objectives

To sum up, the final training objective for a datapoint $x^{(i)}$ can be written in follows:

$$L(x^{(i)},\theta,\phi) = -D_{KL}\left[q_\phi(z|x) \parallel p(z)\right] + \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \tag{15}$$

$$\simeq \frac{1}{2}\sum_{k=1}^{K}\left(1+\ln\sigma_k^2(x;\phi) - \sigma_k^2(x;\phi) - \mu_k^2(x;\phi)\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(x^{(i)}|z^{(i,l)}) \tag{16}$$

- $z^{(i,l)} = \mu(x^{(i)}) + \sigma(x^{(i)}) \odot \varepsilon^{(l)}$ and $\varepsilon \sim \mathcal{N}(0,\mathbb{I})$
- L: Monte Carlo sample size (often $L=1$)
- K: dimensionality of latent variable z
- given a fixed x and ε, Equation 15 is deterministic and continuous in θ and ϕ, so backprop will work.

5. Variational Autoencoder: Additional details

5.1 Sample generation

When we want to generate a new sample, we do not use encoder at test time. Instead we simply input a sample $z \sim \mathcal{N}(0,\mathbb{I})$ into a decoder. Decoder will produce **parameters** of $p(x|z)$, then we sample a new x from $p(x|z)$.

5.2 Manifold learning by VAE

Although the main purpose of VAE is focused on its decoder, it can be used as low-dimensional **manifold learning** because it jointly trains a parametric encoder and decoder.

5.3 Early issue: quality

The main drawback of VAE is a **blurry** image sample, while GAN produces relatively clear output. There is a quality-diversity trade-off in generative learning, and VAE produces more diverse samples sacrificing quality.

5.4 More recent issue: posterior collapse

5.4.1 Background

Deep latent variable models trained with amortized variational inference have led to advances in representation learning on high-dimensional data. It typically has simple decoders because combining simple prior with simple decoder can express more complex distribution (e.g., mapping from z to x using uni-modal Gaussian). This simple decoders are **good** at capturing global structure in input, but **bad** at capturing complex local structure (e.g., texture).

By the way, autoregressive models, such as PixelRNN, show excellent density modeling and sample quality without latent variables. They are**good** at capturing local statistics, but **bad** at globally coherent structures.

Then, why don't we use both of them?

5.4.2 Posterior collapse

A natural idea is combining the power of tractable densities (ARM) and representation learning capabilities (LVM). However, if AR decoder is powerful enough to model the data density, then the model can learn to **ignore** the latent variables. We call it as **posterior collapse**, and it produces a trivial posterior that collapses to the prior: $q_\phi(z|x) \simeq q_\phi(z)$.

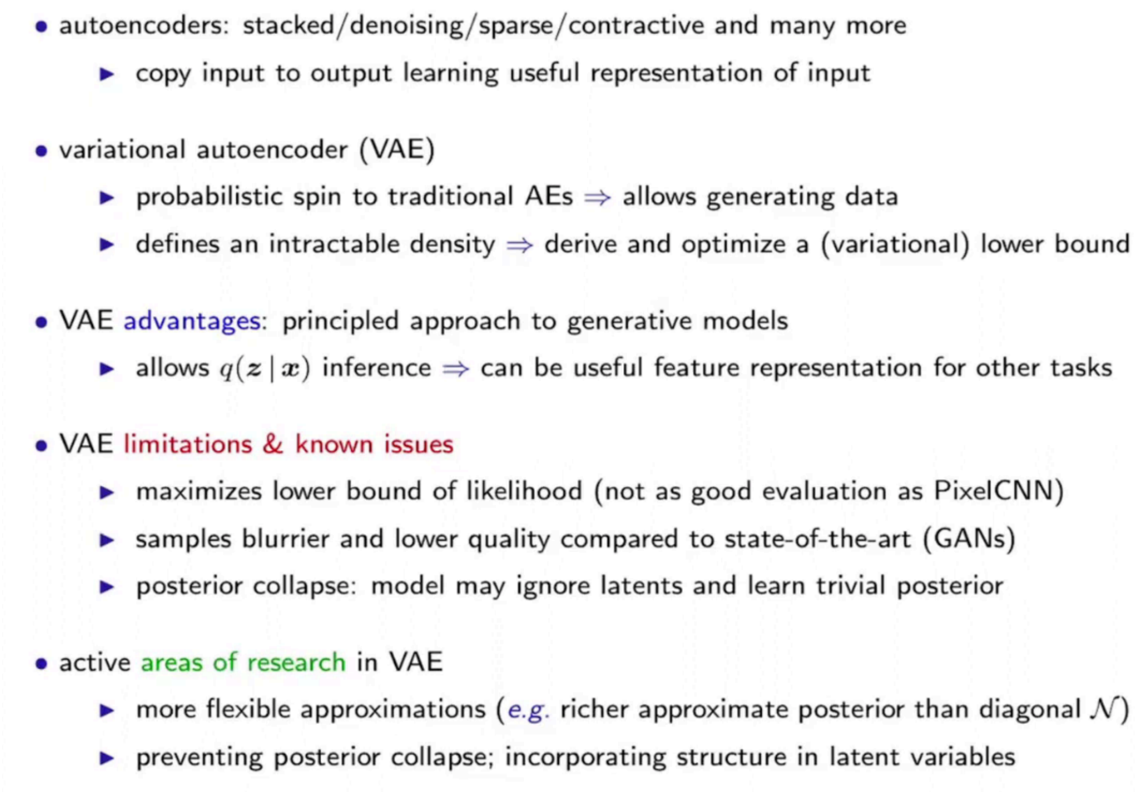
As a result, decoder output $\tilde{x}$ becomes almost **independent** of z. For example, crude representative of all training inputs can be a generic output. To prevent this, there have been studies such as changing ELBO objective to weaken the decoder or carefully selecting variational families (e.g., δ-VAE).

5.5 Notable variants

Recent studies focus on how to structure or contrain latent variables.

- hierarchical: Nouveau VAE
- discrete: VQ-VAE
- disentangled: β-VAE

6. Summary



- autoencoders: stacked/denoising/sparse/contractive and many more
  - copy input to output learning useful representation of input
- variational autoencoder (VAE)
  - probabilistic spin to traditional AEs $\Rightarrow$ allows generating data
  - defines an intractable density $\Rightarrow$ derive and optimize a (variational) lower bound
- VAE advantages: principled approach to generative models
  - allows $q(\boldsymbol{z} \mid \boldsymbol{x})$ inference $\Rightarrow$ can be useful feature representation for other tasks
- VAE limitations & known issues
  - maximizes lower bound of likelihood (not as good evaluation as PixelCNN)
  - samples blurrier and lower quality compared to state-of-the-art (GANs)
  - posterior collapse: model may ignore latents and learn trivial posterior
- active areas of research in VAE
  - more flexible approximations (*e.g.* richer approximate posterior than diagonal $\mathcal{N}$)
  - preventing posterior collapse; incorporating structure in latent variables

Fig. 12. Summary of VAE