Deep Generative Models

Qinliang Su （苏勤亮）

Sun Yat-sen University

suqliang@mail.sysu.edu.cn

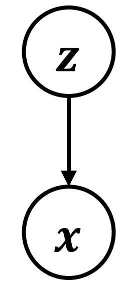
Outline

- Deep Generative Model
- Learning under the VB-EM Framework
- Estimating the Gradient Naively
- Estimating the Gradient using Re-parameterization Trick
- Amortizing the Inference

Generative Models

- Describing the data generation process by a joint pdf

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$$



where $\boldsymbol{x}$ and $\boldsymbol{z}$ denote data and latent variable

Obviously, generative model is a kind of latent-variable models

- $p(\boldsymbol{z})$ and $p(\boldsymbol{x}|\boldsymbol{z})$ are chosen by taking following factors into account

1) Data compatibility 2) Modeling flexibilities 3) Training easiness

- Examples

PCA: $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})$ and $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$

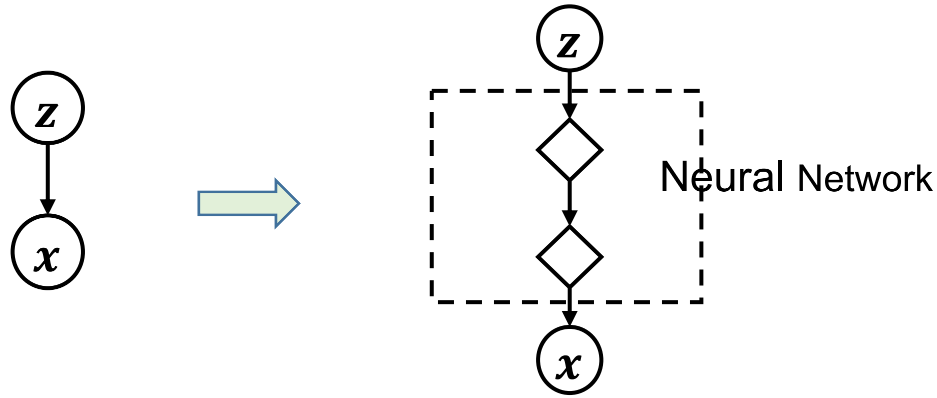
GMM: $p(\boldsymbol{z}) = Cat(\boldsymbol{z}; \boldsymbol{\pi})$ and $p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{m=1}^{M}[\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)]^{z_m}$

Deep Generative Models

- To increase modeling ability, a deep neural network is introduced between $\boldsymbol{z}$ and $\boldsymbol{x}$. Then, the joint pdf becomes

$$p(\boldsymbol{x}, \boldsymbol{z}) = \underbrace{p(\boldsymbol{x} | T(\boldsymbol{z}))}_{p(\boldsymbol{x}|\boldsymbol{z})} \, p(\boldsymbol{z})$$

where $T(\cdot)$ represents a neural network



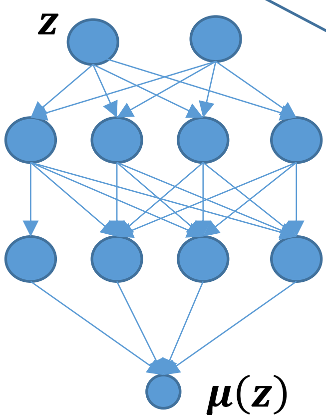
- Comparing to generative models (GM), deep GMs (DGM) let the *conditional pdf rely on a neural-network-transformed variable* $T(\boldsymbol{z})$

- **Example:** To model images, we can specify the joint pdf as

$$p(\boldsymbol{x}, \boldsymbol{z}) = \underbrace{\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}(\boldsymbol{z}), \boldsymbol{I})}_{p(\boldsymbol{x}|\boldsymbol{z})} \underbrace{\mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})}_{p(\boldsymbol{z})}$$

where $\boldsymbol{\mu}(\boldsymbol{z})$ denotes the output of a neural network, *e.g.*,

$$\boldsymbol{\mu}(\boldsymbol{z}) = \boldsymbol{W}_3 a(\boldsymbol{W}_2 a(\boldsymbol{W}_1 \boldsymbol{z} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3$$



In PCA, $\boldsymbol{\mu}(\boldsymbol{z}) = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}$

Denote the parameters in NNs as $\boldsymbol{\theta}$, *i.e.*, $\boldsymbol{\theta} \triangleq \{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=1}^{3}$

Outline

- Deep Generative Model
- Learning under the VB-EM Framework
- Estimating the Gradient Naively
- Estimating the Gradient using Re-parameterization Trick
- Amortizing the Inference

Learning under VB-EM Framework

- DGM is a latent-variable model, hence can be trained with the EM algorithms

Key steps in EM

1) The posteriori distribution $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$
2) Deriving the expectation $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})}[\log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})]$
3) Maximization

- But due to the existence of neural networks, *the exact posterior* $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$ *and expectation* $\mathbb{E}[\cdot]$ are difficult to obtain

- Thus, we resort to VB-EM algorithm by using a simple distribution to approximate the true posterior $p\left(\boldsymbol{z}\middle|\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)$

- VB-EM seeks to maximize the lower bound of log-likelihood

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z}$$

where $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ denotes the approximate posterior

- Substituting $p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ into $\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi})$ gives

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \int_{\boldsymbol{z}} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} - \int_{\boldsymbol{z}} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \ln \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})} d\boldsymbol{z}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - KL\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)$$

- In the subsequent, we consider a concrete example, in which $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ are set as diagonal Gaussian form

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{I}) \qquad q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\lambda}, diag(\boldsymbol{\eta}^2)\right)$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\cdot)$ represents a neural network function with parameter denoted as $\boldsymbol{\theta}$; and $\boldsymbol{\phi} = \{\boldsymbol{\lambda}, \boldsymbol{\eta}\}$ denotes the posterior parameter

- To train the model, we can maximize the lower bound $\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi})$ *w.r.t.* the model and posterior parameter $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. To this end, what we need is the gradient

$$\frac{\partial \mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi})}{\partial \boldsymbol{\theta}} \quad \text{and} \quad \frac{\partial \mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}$$

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - KL\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)$$

- How to obtain the gradient of $\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi})$ *w.r.t.* $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$?

  1) Since $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$ are both Gaussian, the close-form expression of $KL(q_{\boldsymbol{\phi}}||p)$ can be easily obtained, and so does its its gradient

  2) Due to the existence of neural networks, the close-form expression of $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$ cannot be obtained. So, its gradient expression cannot be obtained directly

- Now, the only problem left is how to estimate the derivatives of $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$ *w.r.t.* $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, *i.e.*,

$$\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\theta}} \qquad \frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\phi}}$$

Outline

- Deep Generative Model
- Learning under the VB-EM Framework
- Estimating the Gradient Naively
- Estimating the Gradient using Re-parameterization Trick
- Amortizing the Inference

Estimating $\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]/\partial\boldsymbol{\theta}$

$$\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z}$$

$$= \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \frac{\partial \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})}{\partial \boldsymbol{\theta}} d\boldsymbol{z}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{\partial \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})}{\partial \boldsymbol{\theta}}\right]$$

where $\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = C - \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z})\|^2$, and C is a constant

- Due to the Gaussian form of $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, $\frac{\partial \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})}{\partial \boldsymbol{\theta}}$ can be obtained by using the BP algorithm
- The only obstacle left is about how to evaluate the expectation

$$\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\theta}} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{\partial \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})}{\partial \boldsymbol{\theta}}\right]$$

- Due to the existence of neural network, it is impossible to derive a close-form expression for the expectation, even if $\frac{\partial \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})}{\partial \boldsymbol{\theta}}$ is known
- Instead, we can approximate the expectation with its empirical average, that is,

$$\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\theta}} \approx \frac{1}{K}\sum_{k=1}^{K}\frac{\partial \ln p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|\boldsymbol{z}^{(k)}\right)}{\partial \boldsymbol{\theta}}$$

where $\boldsymbol{z}^{(k)}$ are samples drawn from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$

- From $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\lambda}, diag(\boldsymbol{\eta}^2)\right)$, it is easy to drawn samples from the Gaussian distribution under a given value of $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}^2$, *e.g.*, the parameter value $\boldsymbol{\lambda}_t$ and $\boldsymbol{\eta}_t$ at the t-th step

- In summary, given the model parameter $\boldsymbol{\theta}_t$ and posterior parameter values $\boldsymbol{\lambda}_t$ and $\boldsymbol{\eta}_t$, we can estimate the gradient $\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$ with the following steps

1) Draw Ksamples $\{\boldsymbol{z}^{(k)}\}_{k=1}^{K}$ from the posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\lambda}_t, diag(\boldsymbol{\eta}_t^2)\right)$

2) Use BP to evaluate the gradient $\frac{\partial \ln p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|\boldsymbol{z}^{(k)}\right)}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$ at model parameter $\boldsymbol{\theta}_t$ and latent value $\boldsymbol{z}^{(k)}$

3) Estimate the gradient by empirical average $\frac{1}{K}\sum_{k=1}^{K} \frac{\partial \ln p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|\boldsymbol{z}^{(k)}\right)}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$

Question: To estimate $\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\theta}}$, can we first estimate the expectation $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$ by its emperical average at the given model parameter $\boldsymbol{\theta}_t$ and posterior parameter $\boldsymbol{\lambda}_t$ and $\boldsymbol{\eta}_t$ as

$$\frac{1}{K}\sum_{k=1}^{K} \ln p_{\boldsymbol{\theta}_t}\left(\boldsymbol{x}|\boldsymbol{z}^{(k)}\right),$$

and then compute the gradient of the emperical average w.r.t $\boldsymbol{\theta}$?

- How to estimate the gradient if we want to use this method?
  - Estimate the expression of $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$ for all possibles values of $\boldsymbol{\theta}$

    $$\frac{1}{K}\sum_{k=1}^{K} \ln p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|\boldsymbol{z}^{(k)}\right),$$

    rather than the value only at $\boldsymbol{\theta} = \boldsymbol{\theta}_t$
  - Obviously, it will lead to the same gradient expression as the previous method

Estimating $\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]/\partial \boldsymbol{\phi}$

$$\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\boldsymbol{z}) \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z}$$

$$= \int \frac{\partial q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{\partial \boldsymbol{\phi}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z}$$

$$= \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \frac{\partial \ln q_{\boldsymbol{\phi}}(\boldsymbol{z})}{\partial \boldsymbol{\phi}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{\partial \ln q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{\partial \boldsymbol{\phi}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right]$$

- Similarly, we can also use the empirical average to approximate the expectation, that is,

$$\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\phi}} \approx \frac{1}{K} \sum_{k=1}^{K} \frac{\partial \ln q_{\boldsymbol{\phi}}\left(\boldsymbol{z}^{(k)}|\boldsymbol{x}\right)}{\partial \boldsymbol{\phi}} \ln p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|\boldsymbol{z}^{(k)}\right)$$

Question: Can we first use the samples $\{\boldsymbol{z}^{(k)}\}_{k=1}^{K}$ drawn from the posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\lambda}_t, diag(\boldsymbol{\eta}_t^2)\right)$ to estimate the expression of $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$ w.r.t. $\boldsymbol{\phi}$, and then use the expression to derive the gradient?

- Then answer is no. Why?

- For the method proposed above, it has been observed that

> the variance of the estimated derivative $\frac{1}{K}\sum_{k=1}^{K}\frac{\partial \ln q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{\partial \boldsymbol{\phi}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$
>
> is very large, making the estimate *unreliable*

- Thus, a new method is needed to estimate the derivative $\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\phi}}$

Outline

- Deep Generative Model
- Learning under the VB-EM Framework
- Estimating the Gradient Naively
- Estimating the Gradient using Re-parameterization Trick
- Amortizing the Inference

Re-parameterization Trick

Re-parameterization Trick: For any sample $\boldsymbol{z}^{(i)}$ drawn from the distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\lambda}, diag(\boldsymbol{\eta}^2)\right)$, it can be represented as

$$\boldsymbol{z}^{(i)} = \boldsymbol{\lambda} + \boldsymbol{\eta} \cdot \boldsymbol{\epsilon}^{(i)}$$

where $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_M]$, $\boldsymbol{\eta} = [\eta_1, \cdots, \eta_M]$ and $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\boldsymbol{\epsilon};\ \boldsymbol{0}, \boldsymbol{I})$ (i.e., standard Gaussian noise)

To see this, we can prove that if $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, then the sample $\boldsymbol{z}^{(i)} = \boldsymbol{\lambda} + \boldsymbol{\eta} \cdot \boldsymbol{\epsilon}^{(i)}$ follows the distribution $\mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\lambda}, diag(\boldsymbol{\eta}^2)\right)$, that is,

$$\boldsymbol{z}^{(i)} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})$$

- What is the key differences between Re-parameterization trick and traditional sampling methods?

Remark: We can also use MCMC to draw samples from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\lambda}, diag(\boldsymbol{\eta}^2)\right)$

Samples drawn with the re-parameterized method include the unknown parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$ explicitly, while the traditional sampling methods cannot

- It can be seen that the re-parameterized sample

$$\boldsymbol{z}^{(i)} = \boldsymbol{\lambda} + \boldsymbol{\eta} \cdot \boldsymbol{\epsilon}^{(i)}$$

can separate the parameters from the model randomness

Estimating Derivatives with the Re-parameterization Trick

- Using the re-parameterization trick, the expectation $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$ can be estimated as

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] \approx \frac{1}{K}\sum_{i=1}^{K} \ln p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|\boldsymbol{z}^{(i)}\right)$$

where

$$\boldsymbol{z}^{(i)} = \boldsymbol{\lambda} + \boldsymbol{\eta} \cdot \boldsymbol{\epsilon}^{(i)}$$

- Substituting it into $\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = C - \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z})\|^2$ gives

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] \approx \frac{1}{K}\sum_{i=1}^{K} \left(C - \frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{\lambda} + \boldsymbol{\eta} \cdot \boldsymbol{\epsilon}^{(i)}\right)\right\|^2\right)$$

where $\boldsymbol{\epsilon}^{(i)}$ is a random noise from standard Gaussian $\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \boldsymbol{I})$

- Therefore, the derivatives $\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\phi}}$ can be estimated from the approximate function

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{K}\sum_{i=1}^{K}\left(C - \frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{\lambda} + \boldsymbol{\eta} \cdot \boldsymbol{\epsilon}^{(i)}\right)\right\|^2\right)$$

- That is,

$$\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\theta}} \approx \frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi})}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}{\partial \boldsymbol{\phi}} \approx \frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}$$

- $\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta},\boldsymbol{\phi})}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta},\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}$ can be evaluated with BP algorithm with the automatic tools

Outline

- Deep Generative Model
- Learning under the VB-EM Framework
- Estimating the Gradient Naively
- Estimating the Gradient using Re-parameterization Trick
- Amortizing the Inference

Extending to Large Datasets

- So far, only one training example is considered

- When a training set containing N examples $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^N$ is considered, the training objective becomes

$$\mathcal{L}(\mathcal{X}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{N} \sum_{n=1}^{N} \left(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_n|\boldsymbol{z})] - KL\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)||p(\boldsymbol{z})\right) \right)$$

- To optimize the objective, it is better to let each $\boldsymbol{x}_n$ has its own $\boldsymbol{\phi}_n = \{\boldsymbol{\lambda}_n, \boldsymbol{\eta}_n\}$, that is,

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n) = \mathcal{N}\left(\boldsymbol{z}_n; \boldsymbol{\lambda}_n, diag(\boldsymbol{\eta}_n^2)\right)$$

- With the re-parameterization trick, a sample $\boldsymbol{z}_n^{(k)}$ from $q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)$ can be represented as

$$\boldsymbol{z}_n^{(k)} = \boldsymbol{\lambda}_n + \boldsymbol{\eta}_n \cdot \boldsymbol{\epsilon}_n^{(k)}$$

- Then, the objective $\mathcal{L}(\mathcal{X};\boldsymbol{\theta},\boldsymbol{\phi})$ can be approximated as

$$\mathcal{L} \approx \frac{1}{NK}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(C - \frac{1}{2}\left\|\boldsymbol{x}_n - \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{\lambda}_n + \boldsymbol{\eta}_n \cdot \boldsymbol{\epsilon}_n^{(k)}\right)\right\|^2\right) - \frac{1}{N}\sum_{n=1}^{N} KL\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)||p(\boldsymbol{z}_n)\right)$$

- Training complexity
  - $\boldsymbol{\phi}_n$ is only updated *on* the data sample $\boldsymbol{x}^{(n)}$, while $\boldsymbol{\theta}$ will be updated on all samples
  - Because $\boldsymbol{\phi}_n$ is updated much less frequent than $\boldsymbol{\theta}$, to ensure $q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)$ is a good approximate to true posterior $p_{\boldsymbol{\theta}_t}(\boldsymbol{z}_n|\boldsymbol{x}_n)$, the parameter $\boldsymbol{\phi}_n$ has to be updated much more times than $\boldsymbol{\theta}$

Amortizing the Burden of Inference

- Instead of learning $\boldsymbol{\lambda}_n$ and $\boldsymbol{\eta}_n$ directly, we set $\boldsymbol{\lambda}_n$ and $\boldsymbol{\eta}_n$ as the outputs of neural networks, that is,

$$\boldsymbol{\lambda}_n = g_{\boldsymbol{\phi}_1}(\boldsymbol{x}_n) \qquad\qquad \boldsymbol{\eta}_n = g_{\boldsymbol{\phi}_2}(\boldsymbol{x}_n)$$

where $g_{\boldsymbol{\phi}_\ell}(\cdot)$ represents neural networks parameterized by $\boldsymbol{\phi}_\ell$

- Then, the objective $\mathcal{L}(\mathcal{X};\boldsymbol{\theta},\boldsymbol{\phi})$ can be written as

$$\mathcal{L} \approx \frac{1}{NK}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(C - \frac{1}{2}\left\|\boldsymbol{x}_n - \boldsymbol{\mu}_{\boldsymbol{\theta}}\Big(\underbrace{g_{\boldsymbol{\phi}_1}(\boldsymbol{x}_n) + g_{\boldsymbol{\phi}_2}(\boldsymbol{x}_n)\cdot\boldsymbol{\epsilon}_n^{(k)}}_{\boldsymbol{\lambda}_n + \boldsymbol{\eta}_n \cdot \boldsymbol{\epsilon}_n^{(k)}}\Big)\right\|^2\right)$$

$$-\frac{1}{N}\sum_{n=1}^{N} KL\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)||p(\boldsymbol{z})\big)$$

- Here, the parameters to be optimized are $\{\boldsymbol{\theta},\boldsymbol{\phi}_1,\boldsymbol{\phi}_2\}$

- Differences between the *non-amortized* and *amortized* methods
  - For the non-amortized method, a separate parameter $\{\boldsymbol{\lambda}_n, \boldsymbol{\eta}_n\}$ should be learned for each data example $\boldsymbol{x}_n$
  - For the amortized method, common deep neural networks are learned for all data examples $\{\boldsymbol{x}_n\}_{i=1}^{N}$
- Question: Learning a DNN is much more expensive than learning one $\{\boldsymbol{\lambda}_i, \boldsymbol{\eta}_i\}$. Then, why we choose to learn DNNs for inference?
  - The computation burden of learning a DNN is shouldered by all examples, while the complexity of learning $\{\boldsymbol{\lambda}_n, \boldsymbol{\eta}_n\}$ is undertaken only by the n-th example $\boldsymbol{x}_n$
  - The training complexity amortized on each data sample is low

Examining the Training Process

- By looking at the training objective

noise

$$\mathcal{L} \approx \frac{1}{NK}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(C - \frac{1}{2}\left\|\boldsymbol{x}_n - \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(g_{\boldsymbol{\phi}_1}(\boldsymbol{x}_n) + g_{\boldsymbol{\phi}_2}(\boldsymbol{x}_n)\cdot \boldsymbol{\epsilon}_n^{(k)}\right)\right\|^2\right)$$

decoder encoder

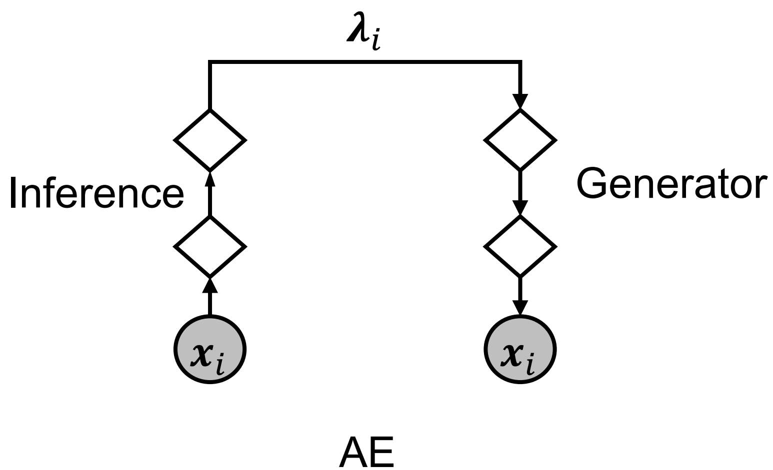
$$-\frac{1}{N}\sum_{n=1}^{N} KL\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)||p(\boldsymbol{z})\right)$$

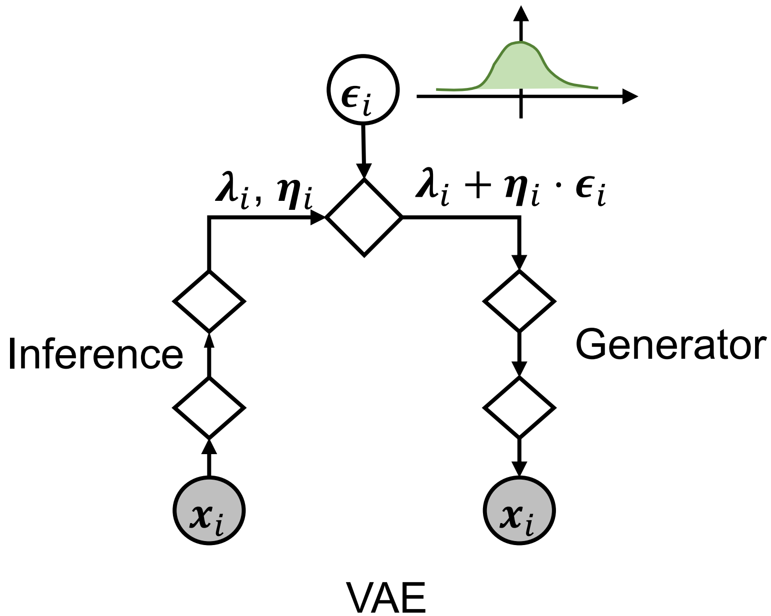
we see that the proposed model is learning an encoder $g_{\boldsymbol{\phi}_\ell}(\cdot)$ and a decoder $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\cdot)$ for reconstruction, with two additional features

1) An additional KL regularizer

2) Imposing noise on the latent code $g_{\boldsymbol{\phi}_1}(\boldsymbol{x}_n)$

- The auto-encoder (AE) is to build an encoder and decoder such that the reconstruction loss is minimized

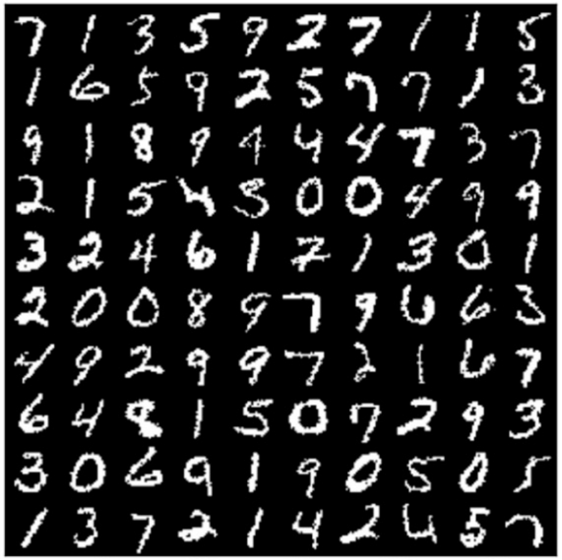




- Comparing to AE, VAE differs in that
  - it includes a regularization term $KL(q||p)$
  - it adds some Gaussian noise in the latent code

Performance

- Samples drawn from VAE trained on MNIST



Samples from training dataset

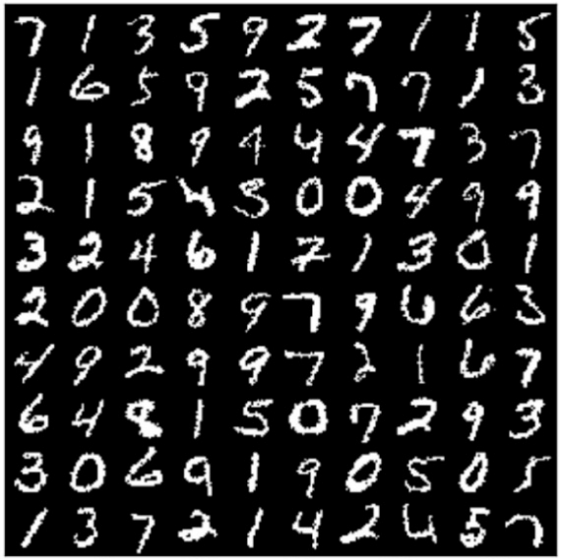
Samples generated by VAE

Improving the Quality of Generated Samples

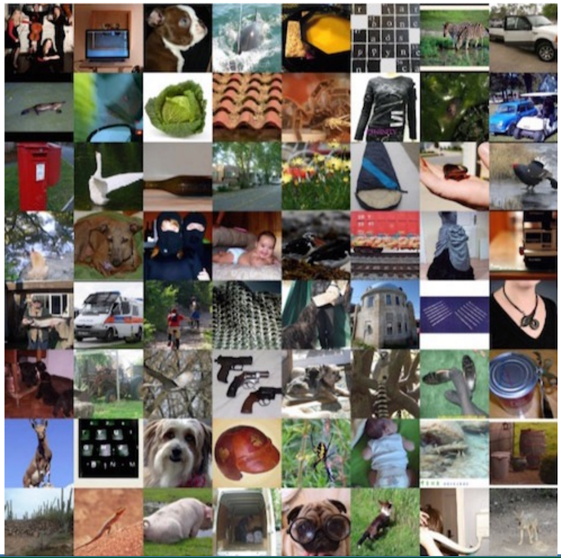
- Improve the inference accuracy
  - Importance weighted auto-encoder (IWAE)
  - Normalizing flow
  - Inverse autoregressive flow
  - Implicit model

- Adversarial training
  - Various GANs…
- Energy-based models
- Diffusion models

Generated Examples



Samples from training dataset



Samples generated by models