The Road from MLE to EM to VAE: A Brief Tutorial

Preprint · June 2020

DOI: 10.13140/RG.2.2.30030.28488/1

Ming Ding
Tsinghua University

The Road from MLE to EM to VAE: A Brief Tutorial

Ming Ding
Department of Computer Science and Technology
Tsinghua University
dm18@mails.tsinghua.edu.cn

Abstract

Variational Auto-Encoders (VAEs) have emerged as one of the most popular genres of *generative models*, which are learned to characterize the data distribution. The classic Expectation Maximization (EM) algorithm aims to learn models with hidden variables. Essentially, both of them are iteratively optimizing the *evidence lower bound* (ELBO) to maximize to the likelihood of the observed data.

This short tutorial joins them up into a line and offer a good way to thoroughly understand EM and VAE with minimal knowledge. It is especially helpful to beginners and readers with experiences in machine learning applications but no statistics background.

1 Introduction

Data can be presumed to be $i.i.d.$ sampled from an underlying distribution in many cases. If we know the data distribution $p^*_{\mathsf{x}}(\cdot)$, we have a command of all the knowledge of them and can generate new data, which is an everlasting pursue of the data scientists.

In practice, we focus on finding a distribution $p_{\mathsf{x}}(\cdot;\theta^*)$ to "best fit" $p^*_{\mathsf{x}}(\cdot)$ in the family $p_{\mathsf{x}}(\cdot;\theta)$ parameterized by θ.[1] The most natural criterion[2] to define "best fit" is *Maximum Likelihood Estimation* (MLE), which maximizes the probability of observed data to be generated.

Sometimes, knowledge about the data encourages us to adopt models with hidden (latent) variables z to approximate the data distribution, where only the joint distribution $p_{\mathsf{x},\mathsf{z}}(\cdot,\cdot;\theta)$ are explicitly defined. The marginal likelihood $p_{\mathsf{x}}(\cdot) = \int_{\mathcal{Z}} p_{\mathsf{x},\mathsf{z}}(\cdot,\mathsf{z};\theta)d\mathsf{z}$ involves an integral over z. Many high-dimensional data, for example images, require complicated deep learning based models, leading to the integral *intractable*[3]. This is the fundamental problem that *Variational Inference* (VI) methods, including Variational Auto-Encoder (VAE) [7], aim to solve.

ELBO is a lower bound of the logarithm of the marginal likelihood $\log p_{\mathsf{x}}(\mathbf{x};\theta)$ and constructed by introducing an extra distribution q_{z}. The closer q_{z} and the posterior $p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta)$ are, the tighter the bound.

The EM algorithm and VAE both iteratively optimize ELBO. More specifically, they alternately optimize ELBO over q_{z} and θ until convergence. The difference is that EM performs a perfect optimization at each step, while VAE performs a gradient step with some other approximation tricks, because it is designed for more complex models, usually neural networks. We will introduce their connections and differences in detail in the following sections.

[1] This is non-Bayesian setting. In Baysian setting, we have priors on the parameter θ and infer a distribution over θ instead of the best one θ^*. We will not discuss it in this tutorial.

[2] In non-Bayesian setting, the maximum likelihood estimator is the *minimum-variance unbiased estimator*, if the latter exists.

[3] The integral has no analytic form or is time-consuming to compute.

2 Notations

Here, we establish some notations to get started.

- Values. We denote $scalars$ using regular serifed fonts, e.g. x, and $\mathbf{vectors}$ using bold fonts, e.g. $\mathbf{x}$. A dataset is usually denoted as $\mathbf{X} = \{\mathbf{x}^{(0)}, ..., \mathbf{x}^{(N)}\}$.
- Random variables. We denote $\mathsf{random\ variables}$ using san-serif fonts, e.g., x. The alphabet of values that x can take on is denoted using caligraphic letter $\mathcal{X}$, i.e. $\mathbf{x} \in \mathcal{X}$.
- Distributions. We denote the probability density (or mass) function of x as $p_{\mathsf{x}}(\cdot)$, which can be abbreviated as p_{x} when no ambiguity is possible. Joint probability density, e.g. $p_{\mathsf{x},\mathsf{y}}(\cdot, \cdot)$, and conditional probability $p_{\mathsf{y}|\mathsf{x}}(\cdot|\cdot)$ are denoted similarly.

Note that we will use $\mathsf{z} \sim q_{\mathsf{z}}$ to denote a random variable whose distribution is q_{z} and alphabet is also $\mathcal{Z}$. It is a different random variable with $\mathsf{z} \sim p_{\mathsf{z}}$ due to different distributions. We do not use another symbol, e.g. z', for the consistency with the existing papers on these topics. Instead, we will annotate the distribution to avoid ambiguity, e.g. $\mathbb{E}_{\mathsf{z}\sim q_{\mathsf{z}}} f(\mathsf{z})$ for the expression of expectation and $\mathrm{KL}\big(q_{\mathsf{z}} \| p_{\mathsf{z}}\big)$ for KL-divergence between q_{z} and p_{z}.

3 Maximum Likelihood Estimation

Suppose we are modeling the data with $p_{\mathsf{x},\mathsf{z}}(\cdot, \cdot; \theta)$, where θ are the parameters of the model and $\mathbf{x}, \mathbf{z}$ are the observed variables and hidden variables.

We want to find the *Maximum Likelihood Estimation* (MLE) $\hat{\theta}_{ML}$ with $i.i.d.$ observed data $\mathbf{X} = \{\mathbf{x}^{(0)}, ..., \mathbf{x}^{(N)}\}$, i.e.

$$\hat{\theta}_{ML}(\mathbf{X}) = \underset{\theta}{\mathrm{argmax}}\, p(\mathbf{X}; \theta),$$

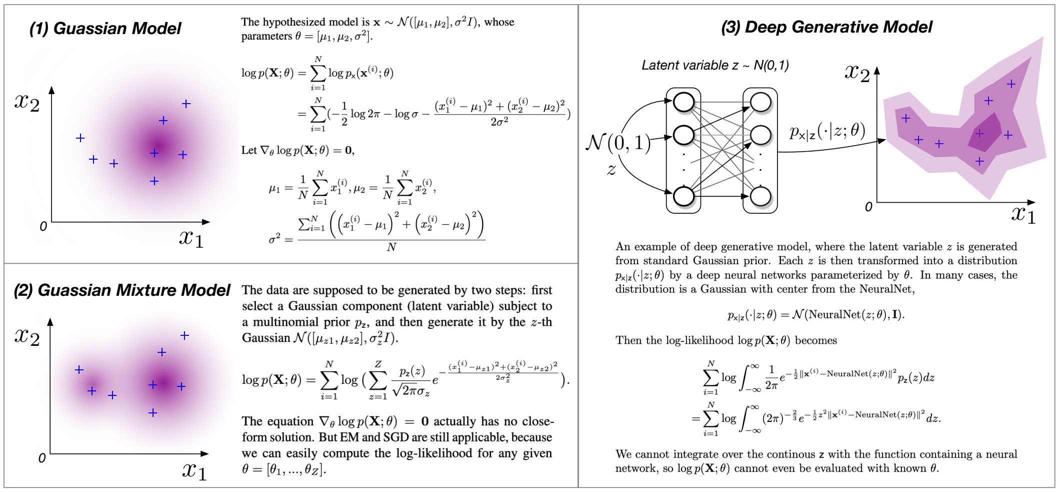
where $p(\mathbf{X}; \theta)$ is the marginal likelihood of X and also called *evidence*. Usually, we calculate the logarithm of evidence to cope with $i.i.d.$ data.

$$\begin{aligned} \log p(\mathbf{X}; \theta) &= \log \prod_{i=1}^{N} p_{\mathsf{x}}(\mathbf{x}^{(i)}; \theta) \\ &= \sum_{i=1}^{N} \log p_{\mathsf{x}}(\mathbf{x}^{(i)}; \theta) \\ &= \sum_{i=1}^{N} \log \int_{\mathcal{Z}} p_{\mathsf{x},\mathsf{z}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}; \theta) d\mathbf{z}^{(i)}. \end{aligned} \tag{1}$$

In this way, the full log-likelihood are decomposed as the summation the log-likelihood of each data point. In the following sections, we will only consider the log-likelihood of one data point $\log p_{\mathsf{x}}(\mathbf{x}; \theta)$ in the analyses, and will still consider multiple data points in the algorithm descriptions.

In fact, the difficulty of MLE varies significantly according to the complexity of our model $p_{\mathsf{x},\mathsf{z}}$. Here, we divide the complexities into three levels from simple to complicated:

1. The equation $\nabla_{\theta} \log p(\mathbf{X}; \theta) = \mathbf{0}$ has close-form solutions $\{\theta^{*(0)}, \theta^{*(1)}, ...\}$. If so, we can evaluate the likelihood $p(\mathbf{X}; \theta^{*(i)})$ at each stationary point $\theta^{*(i)}$ and find the maximum. However, when hidden variables $\mathbf{z}$ exist, $p(\mathbf{X}; \theta)$ involves integrals over $\mathbf{z}$, which usually makes the equation without a close-form solution.
2. Given θ, the marginal likelihood $p(\mathbf{X}; \theta)$ can be evaluated, which also means $\int_{\mathcal{Z}} p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}$ is tractable. In these cases, the most popular and straightforward way nowadays is to leverage automatic differentiation tools, for example Tensorflow and Pytorch, to perform *gradient ascent*, i.e. $\theta \leftarrow \theta + \alpha \nabla_{\theta} p(\mathbf{X}; \theta)$.
   The EM algorithm, which iteratively maximizes the ELBO, is mainly designed for many cases in this class and usually demonstrates robustness and rapid convergence.
3. The marginal likelihood $p(\mathbf{X}; \theta)$ cannot be evaluated, because $\int_{\mathcal{Z}} p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}$ is intractable. This often happens in the "deep learning era" where $p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}; \theta) =$


(1) Guassian Model
x_2
x_1
0
(2) Guassian Mixture Model
x_2
x_1
0
(3) Deep Generative Model
Latent variable z ~ N(0,1)
$\mathcal{N}(0,1)$
z
$p_{\mathsf{x}|\mathsf{z}}(\cdot|z;\theta)$
x_2
x_1
0

The hypothesized model is $\mathbf{x} \sim \mathcal{N}([\mu_1, \mu_2], \sigma^2 I)$, whose parameters $\theta = [\mu_1, \mu_2, \sigma^2]$.

$$\log p(\mathbf{X};\theta) = \sum_{i=1}^{N} \log p_{\mathsf{x}}(\mathbf{x}^{(i)};\theta)$$
$$= \sum_{i=1}^{N} (-\frac{1}{2}\log 2\pi - \log\sigma - \frac{(x_1^{(i)}-\mu_1)^2 + (x_2^{(i)}-\mu_2)^2}{2\sigma^2})$$

Let $\nabla_\theta \log p(\mathbf{X};\theta) = \mathbf{0}$,

$$\mu_1 = \frac{1}{N}\sum_{i=1}^{N} x_1^{(i)}, \mu_2 = \frac{1}{N}\sum_{i=1}^{N} x_2^{(i)},$$
$$\sigma^2 = \frac{\sum_{i=1}^{N}\left(\left(x_1^{(i)}-\mu_1\right)^2 + \left(x_2^{(i)}-\mu_2\right)^2\right)}{N}$$

The data are supposed to be generated by two steps: first select a Gaussian component (latent variable) subject to a multinomial prior p_{z}, and then generate it by the z-th Gaussian $\mathcal{N}([\mu_{z1}, \mu_{z2}], \sigma_z^2 I)$.

$$\log p(\mathbf{X};\theta) = \sum_{i=1}^{N} \log \big(\sum_{z=1}^{Z} \frac{p_{\mathsf{z}}(z)}{\sqrt{2\pi}\sigma_z} e^{-\frac{(x_1^{(i)}-\mu_{z1})^2+(x_2^{(i)}-\mu_{z2})^2}{2\sigma_z^2}}\big).$$

The equation $\nabla_\theta \log p(\mathbf{X};\theta) = \mathbf{0}$ actually has no close-form solution. But EM and SGD are still applicable, because we can easily compute the log-likelihood for any given $\theta = [\theta_1, ..., \theta_Z]$.

An example of deep generative model, where the latent variable z is generated from standard Gaussian prior. Each z is then transformed into a distribution $p_{\mathsf{x}|\mathsf{z}}(\cdot|z;\theta)$ by a deep neural networks parameterized by θ. In many cases, the distribution is a Gaussian with center from the NeuralNet,

$$p_{\mathsf{x}|\mathsf{z}}(\cdot|z;\theta) = \mathcal{N}(\mathrm{NeuralNet}(z;\theta), \mathbf{I}).$$

Then the log-likelihood $\log p(\mathbf{X};\theta)$ becomes

$$\sum_{i=1}^{N} \log \int_{-\infty}^{\infty} \frac{1}{2\pi} e^{-\frac{1}{2}\|\mathbf{x}^{(i)} - \mathrm{NeuralNet}(z;\theta)\|^2} p_{\mathsf{z}}(z) dz$$
$$= \sum_{i=1}^{N} \log \int_{-\infty}^{\infty} (2\pi)^{-\frac{2}{3}} e^{-\frac{1}{2}z^2 \|\mathbf{x}^{(i)} - \mathrm{NeuralNet}(z;\theta)\|^2} dz.$$

We cannot integrate over the continous z with the function containing a neural network, so $\log p(\mathbf{X};\theta)$ cannot even be evaluated with known θ.

Figure 1: Gaussian based examples for the three different complexity levels in MLE.

$p_{\mathsf{z}}(\mathbf{z};\theta) p_{\mathsf{x}|\mathsf{z}}(\mathbf{x}|\mathbf{z};\theta)$ and $p_{\mathsf{x}|\mathsf{z}}(\mathbf{x}|\mathbf{z};\theta)$ is modeled by a neural network. VAE mainly targets at this class.

To better illustrate the difference of difficulties, Figure 1 shows examples of the three levels.

4 The Evidence Lower Bound

The ELBO is the core concept of this tutorial, which is a lower bound of $\log p_{\mathsf{x}}(\mathbf{x};\theta)$ and constructed by introducing an extra distribution q_{z} over the space $\mathcal{Z}$ of the hidden variables. q_{z} can be arbitrarily selected, but affects the tightness of ELBO.

Suppose the observed data point is $\mathbf{x}$, and we can derive ELBO by decomposing $\log p_{\mathsf{x}}(\mathbf{x};\theta)$ as follows:

$$\log p_{\mathsf{x}}(\mathbf{x};\theta) = \int_{\mathcal{Z}} \log \frac{p_{\mathsf{x},\mathsf{z}}(\mathbf{x},\mathbf{z};\theta) q_{\mathsf{z}}(\mathbf{z})}{p_{\mathsf{z}|\mathsf{x}}(\mathbf{z}|\mathbf{x};\theta) q_{\mathsf{z}}(\mathbf{z})} q_{\mathsf{z}}(\mathbf{z}) d\mathbf{z}$$
$$= \mathop{\mathbb{E}}_{\mathsf{z}\sim q_{\mathsf{z}}} [\log \frac{q_{\mathsf{z}}(\mathsf{z})}{p_{\mathsf{z}|\mathsf{x}}(\mathsf{z}|\mathbf{x};\theta)}] - \mathop{\mathbb{E}}_{\mathsf{z}\sim q_{\mathsf{z}}} [\log \frac{q_{\mathsf{z}}(\mathsf{z})}{p_{\mathsf{x},\mathsf{z}}(\mathbf{x},\mathsf{z};\theta)}]$$
$$= \underbrace{\mathrm{KL}\big(q_{\mathsf{z}}(\cdot)\|p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta)\big)}_{\text{Gap}} \underbrace{-\mathrm{KL}\big(q_{\mathsf{z}}(\cdot)\|p_{\mathsf{x},\mathsf{z}}(\mathbf{x},\cdot;\theta)\big)}_{\triangleq\ \mathrm{ELBO}(\theta, q_{\mathsf{z}})} \tag{2}$$

The gap $\mathrm{KL}\big(q_{\mathsf{z}}(\cdot)\|p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta)\big)$, measures the similarity between q_{z} and the posterior of hidden variables $p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta)$.

Another deduction of ELBO is based on *Jensen Inequality*,

$$\log p_{\mathsf{x}}(\mathbf{x};\theta) = \log \int_{\mathcal{Z}} p_{\mathsf{x},\mathsf{z}}(\mathbf{x},\mathbf{z};\theta) d\mathbf{z}$$
$$= \log \int_{\mathcal{Z}} \frac{p_{\mathsf{x},\mathsf{z}}(\mathbf{x},\mathbf{z};\theta)}{q_{\mathsf{z}}(\mathbf{z})} q_{\mathsf{z}}(\mathbf{z}) d\mathbf{z} \qquad \text{(Importance Sampling)}$$
$$\geq \mathop{\mathbb{E}}_{\mathsf{z}\sim q_{\mathsf{z}}} [\log \frac{p_{\mathsf{x},\mathsf{z}}(\mathbf{x},\mathsf{z};\theta)}{q_{\mathsf{z}}(\mathsf{z})}] \qquad \text{(Jensen Inequality)}$$
$$= -\mathrm{KL}\big(q_{\mathsf{z}}(\cdot)\|p_{\mathsf{x},\mathsf{z}}(\mathbf{x},\cdot;\theta)\big). \qquad \text{(ELBO)}$$

Note that ELBO can be further decomposed as follows,

$$\begin{aligned}\mathrm{ELBO}(\theta, q_{\mathsf{z}}) &= \mathop{\mathbb{E}}_{\mathsf{z}\sim q_{\mathsf{z}}}[\log p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathsf{z}; \theta)] - \underbrace{\mathop{\mathbb{E}}_{\mathsf{z}\sim q_{\mathsf{z}}}[\log q_{\mathsf{z}}(\mathsf{z})]}_{\text{The Entropy of } q_z} && (3)\\ &= \underbrace{\mathop{\mathbb{E}}_{\mathsf{z}\sim q_{\mathsf{z}}}[\log p_{\mathsf{x}|\mathsf{z}}(\mathbf{x}|\mathsf{z}; \theta)]}_{-\text{ reconstruction loss}} - \underbrace{\mathrm{KL}\big(q_{\mathsf{z}}(\cdot)\|p_{\mathsf{z}}(\cdot;\theta)\big)}_{\text{KL between } q_z \text{ and prior}}. && (4)\end{aligned}$$

ELBO provides a new perspective to find or approximate the maximum likelihood,

$$\begin{aligned}\log p_{\mathsf{x}}(\mathbf{x}; \theta) &= \max_{q_{\mathsf{z}}} \max_{\theta} \mathrm{ELBO}(\theta, q_{\mathsf{z}}) && (5)\\ &\approx \max_{q_{\mathsf{z}} \in \{q_{\mathsf{z}}(\cdot;\phi)\}} \max_{\theta} \mathrm{ELBO}(\theta, q_{\mathsf{z}}). && (6)\end{aligned}$$

The Equation (5) illustrates how EM maximizes the likelihood. The Equation (6) says that we can choose q_{z} from a simple family to approximate the likelihood, and meanwhile keep ELBO easy to compute. This is the main idea of VI.

5 The Expectation-Maximization Algorithm

The EM algorithm has been successfully applied on learning many famous models [1], for example GMM and HMM, and is seen as one of the most important algorithms in the 20th century [11].

The EM algorithm is the *coordinate ascent* on $\mathrm{ELBO}(\theta, q_{\mathsf{z}})$.

(E-step) First, we optimize $\mathrm{ELBO}(\theta, q_{\mathsf{z}})$ over q_{z} with fixed θ. According to Eq. (2),

$$\mathrm{ELBO}(\theta, q_{\mathsf{z}}) = \underbrace{\log p_{\mathsf{x}}(\mathbf{x}; \theta)}_{\text{unrelated to } q_z} - \mathrm{KL}\big(q_{\mathsf{z}}(\cdot)\|p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta)\big),$$

$$q_{\mathsf{z}}^*(\theta) \triangleq \operatorname*{argmax}_{q_{\mathsf{z}}} \mathrm{ELBO}(\theta, q_{\mathsf{z}}) = p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta).$$

(M-step) Next, we will optimize over θ with fixed q_{z}. Suppose the current (old) parameter is θ' and plug $q_{\mathsf{z}} = q_{\mathsf{z}}^*(\theta') = p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta')$ into $\mathrm{ELBO}(\theta, q_{\mathsf{z}})$, and we aim to find

$$\theta^* \triangleq \operatorname*{argmax}_{\theta} \mathrm{ELBO}(\theta, p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta')).$$

Recall the decomposition in Eq. (3),

$$\mathrm{ELBO}(\theta, p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta')) = \mathop{\mathbb{E}}_{\mathsf{z}\sim p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta')}[\log p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathsf{z}; \theta)] + \underbrace{H(p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta'))}_{\text{unrelated to } \theta},$$

where $H(p)$ represents the entropy of a distribution p. If we define

$$U(\theta;\theta') \triangleq \mathop{\mathbb{E}}_{\mathsf{z}\sim p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta')}[\log p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathsf{z}; \theta)],$$

the optimal (new) becomes $\theta^* = \operatorname{argmax}_{\theta} U(\theta;\theta')$. The two steps are repeated until convergence. The whole process is summarized in Algorithm 1.

According to Xu and Jordan [12], the advantages of EM algorithm over gradient ascent includes monotonic convergence (see Appendix A), low computational overhead and remarkable performance for some important models. It also automatically satisfies probabilistic constraints. However, EM requires the posterior $p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta')$ tractable and $U(\theta;\theta')$ friendly to maximization, which are rare in complicated models.

6 Variational EM, MCEM and Generalized EM

In this section, we will briefly introduce how to perform EM approximately for the models violating EM's requirements.

- When $p(\mathbf{X};\theta) = \int_{\mathcal{Z}} p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}$ is intractable, the posterior $p_{\mathsf{z}|\mathsf{x}}(\mathbf{z}|\mathbf{x};\theta')$ cannot be evaluated. [4] An alternative solution is *Variational EM*, which replaces the posterior by $q_{\mathsf{z}} \in$

[4] Actually pure EM algorithm only needs $U(\theta;\theta')p_{\mathsf{x}}(\mathbf{x};\theta) = \int_{\mathcal{Z}} p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}; \theta') \log p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}$ to be tractable, which is in general at the same difficulty level as the tractability of $p(\mathbf{X};\theta)$.

Algorithm 1 EM algorithm

```
**Input:** The observed data $\mathbf{X} = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}\}$
Initialize the estimated parameter $\theta'$ at random.
**repeat**
    $U(\theta; \theta') = \sum_{i=1}^{N} \mathbb{E}_{\mathsf{z}^{(i)} \sim p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x}^{(i)};\theta')} [\log p(\mathbf{x}^{(i)}, \mathsf{z}^{(i)}|\theta)]$// Simplify $U(\theta; \theta')$ into an analytic function of $\theta$
    $\theta' \leftarrow \arg\max_{\theta} U(\theta; \theta')$
**until** convergence.
```

a family of simple distributions. For instance, in the mean field method, components on each dimension are independent, i.e.,

$$q_{\mathsf{z}}(\mathbf{z}) = \prod_{i=1}^{\dim \mathbf{z}} q_z(\mathbf{z}_i).$$

- When $U(\theta; \theta') = \mathbb{E}_{\mathsf{z} \sim p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x};\theta')}[\log p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}; \theta)]$ cannot be simplified into an analytic form, we can make a Monte Carlo approximation, resulting in *MCEM*,

  $$U(\theta; \theta') \simeq \frac{1}{M} \sum_{i=1}^{M} \log p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}^{(i)}; \theta),$$

  where $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(M)}\}$ are sampled from $p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x}; \theta')$.
- When the maximum of $U(\theta; \theta')$ cannot be directly got, *Generalized EM* instead seeks a step to increase the ELBO, for example a gradient step.

7 Variational Auto-Encoder

Suppose we want to infer the parameters θ in such a model:

1. $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
2. $\mathbf{x} \sim \mathcal{N}(Decoder(\mathbf{z}; \theta), \mathbf{I})$, $Decoder$ is a Neural Net.

This is the hardest example in Figure 1. Due to the existence of the Neural Net, we are faced with all the three conditions in section 6. If we combine the three mentioned solutions (Variational EM, MCEM and Generalized EM), we will reach VAE! Next, we will discuss how VAE is viewed as an extension of EM.

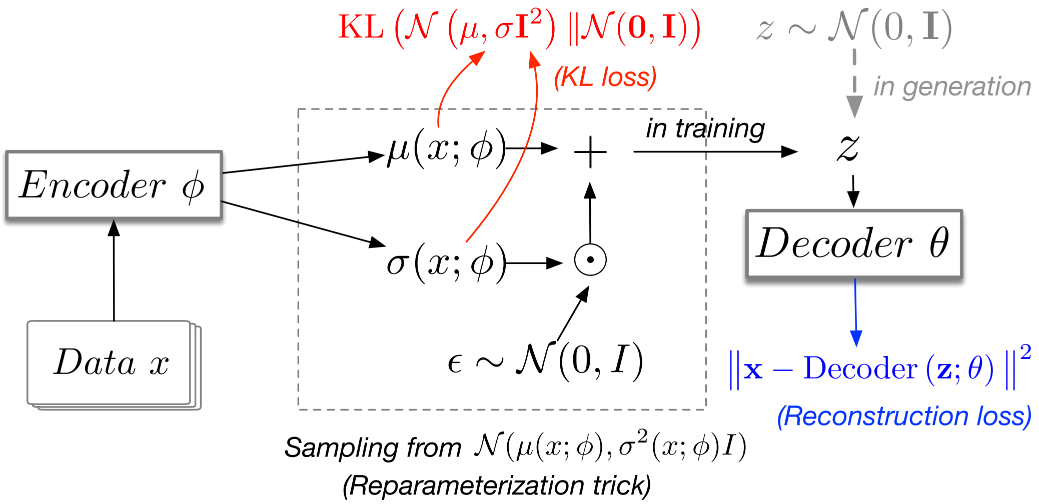
Connection to Variational EM. The q_{z} in VAE is limited to the family of isotropic Gaussian distributions, whose mean and variance can be generated by another neural network $Encoder$, i.e.,

$$\begin{aligned} q_{\mathsf{z}}(\cdot) &= \mathcal{N}(\mu, \sigma^2 I), \\ \mu, \sigma &= Encoder(\mathbf{x}; \phi), \end{aligned}$$

where μ, σ are vectors. In traditional variational EM, we should find the best $q_{\mathsf{z}} = \mathcal{N}(\mu_{\mathbf{x}}, \sigma^2_{\mathbf{x}} I)$ to maximize ELBO(θ, q_{z}) for each observed data point $\mathbf{x}$. VAE uses a trick called *Amortized Variational Inference* (AVI), where $\mu_{\mathbf{x}}, \sigma^2_{\mathbf{x}}$ are outputs of an encoder whose parameters ϕ are *shared* among different data points $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}$. AVI sacrifices some space of q_{z} for training efficiency.

Connection to MCEM and Generalized EM. Generalized EM tells us that we do not need to maximize ELBO in each E-step or M-step. To optimize θ and ϕ via SGD is also acceptable, though cost more steps than vanilla EM. According to Equation (4),

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{\mathsf{z} \sim q_{\mathsf{z}}(\cdot;\phi)}[\log p_{\mathsf{x}|\mathsf{z}}(\mathbf{x}|\mathbf{z}; \theta)] - \text{KL}\big(q_{\mathsf{z}}(\cdot; \phi) \| \mathcal{N}(0, \mathbf{I})\big)$$



$\mathrm{KL}\left(\mathcal{N}\left(\mu, \sigma \mathbf{I}^2\right) \| \mathcal{N}(\mathbf{0}, \mathbf{I})\right)$
(KL loss)
$z \sim \mathcal{N}(0, \mathbf{I})$
in generation
$Encoder\ \phi$
$\mu(x;\phi)$
$+$
in training
z
$\sigma(x;\phi)$
$\odot$
$Decoder\ \theta$
$Data\ x$
$\epsilon \sim \mathcal{N}(0, I)$
$\left\|\mathbf{x} - \mathrm{Decoder}\left(\mathbf{z};\theta\right)\right\|^2$
(Reconstruction loss)
Sampling from $\mathcal{N}(\mu(x;\phi), \sigma^2(x;\phi)I)$
(Reparameterization trick)

Figure 2: Variational Auto-Encoder. In the training of encoder ϕ and decoder θ, the latent variable z are sampled from posterior $\mathcal{N}\left(\mu(x;\phi), \sigma^2(x;\phi)I\right)$, while from prior $\mathcal{N}(0, \mathbf{I})$ in generation (gray). The KL loss $\mathcal{L}_{KL}$ and reconstruction loss $\mathcal{L}_{recon}$ are denoted by red and blue colors respectively.

Algorithm 2 Variational Auto-Encoder

```
Input: The observed data $\mathcal{D} = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}\}$.
    Batch size $n$, learning rate $\alpha$, sampling size $M$.
Initialize the parameters $\theta, \phi$ at random.
repeat
    Select a batch $\mathbf{X} = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(n)}\}$ from $\mathcal{D}$ at random.
    for each $\mathbf{x}^{(i)} \in \mathbf{X}$ do
        Generate noise $\{\epsilon^{(1)}, ..., \epsilon^{(M)}\}$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.
        $\mu, \sigma \leftarrow Encoder(\mathbf{x}^{(i)}; \phi)$.
        for $l \leftarrow 1$ to $M$ do
            $\mathbf{z}^{(l)} \leftarrow \mu + \sigma \odot \epsilon^{(l)}$.
        end for
        $\mathcal{L}^{(i)}_{recon} \leftarrow \frac{1}{2M} \sum_{l=1}^{M} \left\|\mathbf{x}^{(i)} - Decoder(\mathbf{z}^{(l)}; \theta)\right\|^2$.
        $\mathcal{L}^{(i)}_{KL} \leftarrow -\frac{1}{2} \sum_{j=1}^{\dim \mathbf{z}} \left(1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2\right)$.
    end for
    $\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{i=1}^{n} \mathcal{L}^{(i)}_{recon}$.
    $\phi \leftarrow \phi - \alpha \nabla_\phi \sum_{i=1}^{n} (\mathcal{L}^{(i)}_{recon} + \mathcal{L}^{(i)}_{KL})$.
until convergence.
```

Then we optimize $\mathrm{ELBO}(\theta, \phi)$ by gradient ascent,

$$\begin{aligned}
\nabla_\theta \mathrm{ELBO}(\theta, \phi) &= \nabla_\theta \left(\mathop{\mathbb{E}}_{\mathsf{z} \sim q_{\mathsf{z}}(\cdot;\phi)} [\log p_{\mathsf{x}|\mathsf{z}}(\mathbf{x}|\mathbf{z};\theta)] - \mathrm{KL}\big(q_{\mathsf{z}}(\cdot;\phi) \| \mathcal{N}(0, \mathbf{I})\big) \right) \\
&\simeq \nabla_\theta \frac{1}{M} \sum_{i=1}^{M} \log p_{\mathsf{x},\mathsf{z}}(\mathbf{x}, \mathbf{z}^{(i)}; \theta) \\
&= \nabla_\theta \frac{1}{M} \sum_{i=1}^{M} \log e^{-\frac{1}{2}\|\mathbf{x} - Decoder(\mathbf{z}^{(i)})\|^2} \\
&= -\nabla_\theta \underbrace{\frac{1}{2M} \sum_{i=1}^{M} \|\mathbf{x} - Decoder(\mathbf{z}^{(i)}; \theta)\|^2}_{\triangleq \mathcal{L}_{recon}}.
\end{aligned}$$

The gradients of θ are directly got by a back-propagation from *reconstruction loss* $\mathcal{L}_{recon}$, the mean-square error between observed data $\mathbf{x}$ and models' output $Decoder(\mathbf{z}; \theta)$.

However, $\mathbf{z}^{(i)}$ cannot be seen as constant when computing the gradients of ϕ, because $\mathbf{z}^{(i)} \sim \mathcal{N}(\mu(\mathbf{x};\phi), \sigma(\mathbf{x};\phi)^2 I)$, i.e., the sampling distribution is dependent on ϕ.

The solution is *reparameterization*[5], projecting samples from an unrelated distribution, e.g. $\mathcal{N}(\mathbf{0}, \mathbf{I})$, to samples from target distribution $\mathcal{N}(\mu, \sigma^2\mathbf{I})$ by differentiable operations. In VAE, each $\mathbf{z}^{(i)} \sim \mathcal{N}(\mu(\mathbf{x};\phi), \sigma(\mathbf{x};\phi)^2\mathbf{I})$ are generated by $\mathbf{z}^{(i)} = \mu(\mathbf{x};\phi) + \sigma(\mathbf{x};\phi)\epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.

$$\begin{aligned}\nabla_\phi \text{ELBO}(\theta, \phi) &= \nabla_\phi \left(\mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} [\log p_{\mathsf{x}|\mathsf{z}}(\mathbf{x}|\mu(\mathbf{x};\phi) + \sigma(\mathbf{x};\phi)\epsilon; \theta)] - \underbrace{\text{KL}\big(q_{\mathsf{z}}(\cdot;\phi) \| \mathcal{N}(0, \mathbf{I})\big)}_{\triangleq \mathcal{L}_{KL}} \right) \\ &= -\nabla_\phi(\mathcal{L}_{recon} + \mathcal{L}_{KL}).\end{aligned}$$

$\mathcal{L}_{KL}$ is the KL-divergence of two isotropic Gaussian distributions, whose analytic solution is

$$\text{KL}\big(\mathcal{N}(\mu, \sigma\mathbf{I}^2) \| \mathcal{N}(\mathbf{0}, \mathbf{I})\big) = -\frac{1}{2} \sum_{j=1}^{\dim \mathbf{z}} \left(1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2\right).$$

Now we have introduced how VAE infers the parameters of a neural-network decoder and the motivations. The complete training process are illustrated in Algorithm 2 and Figure 2.

8 Advance topics about VAE

8.1 Disentangling in VAE

The most notable difference between VAE and ordinary auto-encoder [6] is that the latent variable z has definite *prior*. To minimize $\mathcal{L}_{KL} = \text{KL}\big(q_{\mathsf{z}}(\cdot;\phi) \| p_{\mathsf{z}}\big)$ limits the capacity of z, which, combined with the pressure to maximize the log likelihood of the training data x under the model, should encourage the model to learn the most efficient representation of the data, i.e. encouraging z to disentangle on different dimensions. A simple modification, β-VAE, introduces a scaling factor $\beta > 1$ for $\mathcal{L}_{KL}$ to upweight the importance of disentangling and was sota method for disentangling for a time [5].

8.2 Forward vs. reverse KL divergence

The MLE-based generative models in this tutorial, are actually minimizing the *forward KL divergence* $\text{KL}\big(p_{\mathsf{x}}^* \| p_{\mathsf{x}}(\cdot;\theta)\big)$. A disadvantage of these methods is the poor fidelity if we generate samples (e.g. images) from the model, because they only enforces the model to assign high likelihoods for real samples, but neglects the regions in $\mathcal{Z}$ with low $p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x})$. $\mathbf{z}$ sampled from these regions could still be mapped to $\mathbf{x}$ with low $p_{\mathsf{x}}^*(\mathbf{x})$. Many recent works are successful to improve the fidelity, for example VQ-VAE [9].

Generative adversarial net (GAN) [4] is another popular generative model, which minimizes the JS-divergence via adversarial learning. $\text{JS}\big(p, q\big) \triangleq \frac{1}{2}\text{KL}\big(p \| \frac{1}{2}(p+q)\big) + \frac{1}{2}\text{KL}\big(q \| \frac{1}{2}(p+q)\big)$ is a mixture of forward and reverse KL. In practice, the generator cannot be optimized perfectly in function space, and thus the models cares more about the reverse KL. The reverse KL loss tends to generate samples only in regions with high $p_{\mathsf{x}}^*(\mathbf{x})$, making the generated samples more realistic than forward KL, but it also leads to the *mode collapse* phenomenon, i.e., the generation cannot cover all the modes of data distribution. Nowozin et al. [8] presented more detailed analyses about this topic.

9 Conclusion

This tutorial has focused on MLE-based generative models, paving the road from MLE to EM to VAE. The prime aim of this tutorial is to provide an ease way to thoroughly understand VAE with minimal prior knowledge, thus many valuable topics are omitted without deep discussions, such as variational inference, Monte Carlo methods, relationships between VAE and other generative models, etc. This tutorial is expected to be the basis for beginners to further investigate these topics.

[5] *Score function estimator*, $\nabla_\theta \mathop{\mathbb{E}}_{x\sim p_{\mathsf{x}}(\cdot;\theta)}[f(x)] = \mathop{\mathbb{E}}_{x\sim p_{\mathsf{x}}(\cdot;\theta)}[f(x)\nabla_\theta \log p_{\mathsf{x}}(x;\theta)]$ is another solution, which has a much higher variance.

Acknowledgments

First I thank Prof. Yuan Shen in Tsinghua University who taught me the basics of statistical inference, and my advisor Jie Tang who always tells me to help other researchers, no matter beginners or experts. I welcome discussions via emails on the tutorial, my researches on graph representation learning, for example GraphSGAN [2], ProNE [14] and markov chain negative sampling [13], and recent interest on cognitive AI, for example CogQA [3].

References

[1] J. A. Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[2] M. Ding, J. Tang, and J. Zhang. Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 913–922, 2018.

[3] M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703, 2019.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2 (5):6, 2017.

[6] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[7] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[8] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[9] A. Razavi, A. v. d. Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

[10] C. J. Wu et al. On the convergence properties of the em algorithm. *The Annals of statistics*, 11 (1):95–103, 1983.

[11] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14 (1):1–37, 2008.

[12] L. Xu and M. I. Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural computation*, 8(1):129–151, 1996.

[13] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou, and J. Tang. Understanding negative sampling in graph representation learning. *arXiv preprint arXiv:2005.09863*, 2020.

[14] J. Zhang, Y. Dong, Y. Wang, J. Tang, and M. Ding. Prone: fast and scalable network representation learning. In *Proc. 28th Int. Joint Conf. Artif. Intell., IJCAI*, pages 4278–4284, 2019.

Appendix

A The proof of the convergence of the EM algorithm

Proof. Suppose θ' is updated from θ_0 to θ_1. Since θ_1 is the maximizer of $\mathbb{E}_{\mathsf{z}\sim p_{\mathsf{z}|\mathsf{x}}(\cdot|\mathbf{x},\theta_0)}[\log p(\mathbf{x},\mathsf{z}|\theta)]$,

$$\mathop{\mathbb{E}}_{\mathsf{z}\sim p(\mathsf{z}|\mathbf{x},\theta_0)}[\log p(\mathbf{x},\mathsf{z}|\theta_1)] \geq \mathop{\mathbb{E}}_{\mathsf{z}\sim p(\mathsf{z}|\mathbf{x},\theta_0)}[\log p(\mathbf{x},\mathsf{z}|\theta_0)]. \tag{7}$$

Meanwhile, according to *Gibbs Inequality*[6],

$$\mathop{\mathbb{E}}_{\mathsf{z}\sim p(\mathsf{z}|\mathbf{x},\theta_0)}[-\log p(\mathsf{z}|\mathbf{x},\theta_1)] \geq \mathop{\mathbb{E}}_{\mathsf{z}\sim p(\mathsf{z}|\mathbf{x},\theta_0)}[-\log p(\mathsf{z}|\mathbf{x},\theta_0)].$$

$$\begin{aligned}
\log p(\mathbf{x}|\theta_1) &= \log \frac{p(\mathbf{x},\mathsf{z}|\theta_1)}{p(\mathsf{z}|\mathbf{x},\theta_1)} \\
&= \mathop{\mathbb{E}}_{\mathsf{z}\sim p(\mathsf{z}|\mathbf{x},\theta_0)}[\log \frac{p(\mathbf{x},\mathsf{z}|\theta_1)}{p(\mathsf{z}|\mathbf{x},\theta_1)}] \\
&\geq \mathop{\mathbb{E}}_{\mathsf{z}\sim p(\mathsf{z}|\mathbf{x},\theta_0)}[\log \frac{p(\mathbf{x},\mathsf{z}|\theta_0)}{p(\mathsf{z}|\mathbf{x},\theta_0)}] \\
&= \log p(\mathbf{x}|\theta_0).
\end{aligned}$$

□

The EM algorithm is monotonic convergence, and if $U_{\theta|\theta'}(\cdot|\cdot)$ is continuous in both θ and θ', EM will convergence to a stationary point θ^* of $\log p(\mathbf{X};\theta)$ [10].

[6] https://en.wikipedia.org/wiki/Gibbs_inequality