第六讲

简介：

和上两讲的内容不同，本讲着重于开发一个对于 Generative model 的全新思路：Latent Variable Model。通过数学变形的技巧，建立起了一个 iterative training process。从单个数据点到多个数据点，model 成为了所谓 Variational Autoencoder，其 loss function 也被高斯分布等 computational efficient 的设计所简化。最后，探讨了 Variational Autoencoder 的各个方面和潜在的不足，以及发展空间。

目录：

Latent Variable Model: Introduction

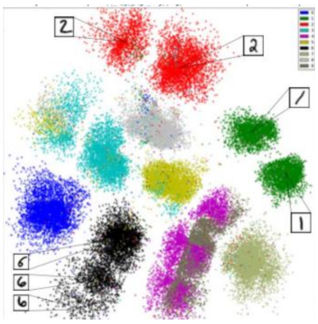
让我们暂时从前面几讲的各种 generative model 中离开，转而重新考虑我们的问题：一般的 generative model 的目的是学习一个分布$p(x)$，其中x对应着数据。为了使得模型可以学习这个分布，**Latent Variable Model** 引入所谓 **latent variable** z：比如说，如果x是人的图片，那么z就可能是人的性别、头发和眼睛颜色等因素，这些分布有助于生成x。我们有

$$p(x,z) = p(z)p(x|z)$$

这里$p(z)$对应着因素的分布，在上面的例子里就是男女比例、头发颜色占比等。$p(x|z)$就是用我们的模型给出。比如说，我们可以取

$$p(x|z) = N\big(\mu(z), \Sigma(z)\big)$$

一个例子是 **Gaussian Mixture Model**，它取$z \sim \mathrm{Catorgical}(w_1, \dots, w_k)$，也就是说$z$有一定概率分别取$w_1, \dots, w_k$；而$p(x|z)$对于每个$z$都是高斯分布。这样的分布在图上画出来就像是若干高斯分布的团块的并，因此叫做“Gaussian Mixture”。



我们接下来考虑这一模型该如何训练。我们首先把注意力集中到一个数据点x上，我们的目的是使得

$$L(x;\theta) = \log p(x) = \log\left(\sum_z p(x,z;\theta)\right)$$

最大。但是这里出现了和之前 sampling 一样的问题：这个求和$\sum_z p(x,z;\theta)$不容易计算。因此我们决定给出一个近似：

$$L(x;\theta) = \log\left(\sum_z q(z)\cdot\frac{p(x,z;\theta)}{q(z)}\right) \geq \sum_z q(z)\log\left(\frac{p(x,z;\theta)}{q(z)}\right)$$

这里我们使用了琴生不等式，因此右边得到的是 loss 函数的一个下界，也被叫做 **ELBO（Evidence Lower Bound）**。可以发现，当$q(z) = p(z|x;\theta)$的时候，这里的近似取得等号，这也就是我们需要达到的目标。这样，我们就给出了一个训练的过程：

Iterative training process

1. 固定$q(z)$为 proposal distribution，训练θ使得最大化

$$L(x;\theta) = \sum_z q(z)\log\left(\frac{p(x,z;\theta)}{q(z)}\right)$$

2. 得到θ之后，替换$q(z) = p(z|x;\theta)$。这一操作是为了在后面的训练过程中让q更加精确。

3. 重复上面的操作，直到到达最值。

但是这个过程中仍然存在一定的问题：我们应该如何做出上面红色部分的替换呢？如果直接划等号就违背了我们认为$q(z)$很好 sample 的原则；因此，我们应该**训练**

另外一个模型$\boldsymbol{q(z;\phi)}$，专门用来模拟$p(z|x;\theta)$。而这个训练的 loss 就对应着 **KL divergence**：

$$KL(q||p) = \sum_z q(z;\phi) \log \frac{q(z;\phi)}{p(z|x;\theta)}$$

这个过程也被叫做 **Variational Inference**。需要注意这里我们使用的叫做 **reverse KL**，也就是把q放在前面，事实上可以证明这样的写法具有更好的数学性质。这样，我们的训练过程就变成了

Iterative training process

1. 固定$q(z;\phi)$为 proposal distribution，训练θ使得最大化 ELBO

$$L(x;\theta) = \sum_z q(z;\phi) \log\left(\frac{p(x,z;\theta)}{q(z;\phi)}\right)$$

2. 得到θ之后，训练ϕ使得最小化

$$KL(q||p) = \sum_z q(z;\phi) \log \frac{q(z;\phi)}{p(z|x;\theta)}$$

3. 重复上面的操作，直到到达最值。

让我们继续研究这个 **Variational Inference** 过程，改写

$$KL(q||p) = \log p(x;\theta) - \sum_z q(z;\phi) \log \frac{p(z,x;\theta)}{q(z;\phi)}$$

我们忽然发现第一项是常数，而第二项就是刚才的 ELBO！我们还可以看到，恰好有

$$\log p(x;\theta) = \sum_z q(z;\phi) \log \frac{p(z,x;\theta)}{q(z;\phi)} + KL(q||p) = \mathrm{ELBO} + KL(q||p)$$

这样这一方法的思路就清晰了：ELBO 对应的是对$p(x;\theta)$的一个估计，而 KL divergence 对应着的是我们的估计和真实的$p(x;\theta)$之间的 approximation error。

在之前的讨论中，我们只考虑了对一个数据x进行训练的情况；而实际上我们有一整个数据集的数据，也就对应这很多的x。上面的训练中，我们需要一个$q(z;\phi)\approx p(z,x;\theta)$ 的分布，但是对于多个x我们却不能对于每一个x单独训练一个q_x，因为这样开销太大了。解决办法是所谓 **Amortized Variational Inference**：我们给出一个完整的$q(z|x;\phi)$，并把所有的x放在一起训练。这样，Variational Inference 对应的目标就变成了

$$J(\theta,\phi;x)=\sum_z q(z|x;\phi)\log\frac{p(z,x;\theta)}{q(z|x;\phi)}\left(\Leftrightarrow\sum_z q(z|x;\phi)\log\frac{p(z|x;\theta)}{q(z|x;\phi)}\right)$$

（这里的等价符号代表着两个目标函数只相差一个不依赖于ϕ的常数）回想以下之前的时候 Variational Inference 对应的目标函数可以写为$\log p(x)$和 KL divergence 之差，我们可以料想现在也可以写出类似的表达式。实际上也确实是这样：

$$J(\theta,\phi;x)=\sum_z q(z|x;\phi)\log p(x|z;\theta)+\sum_z q(z|x;\phi)\log\frac{p(z;\theta)}{q(z|x;\phi)}$$

$$=E_{z\sim q(z|x;\phi)}[\log p(x|z;\theta)]-KL\big(q(z|x;\phi)||p(z;\theta)\big)$$

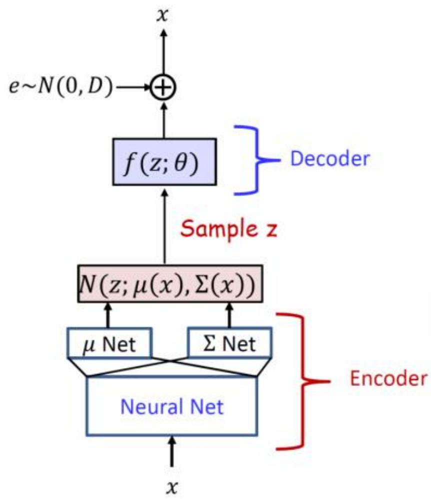
其中第一项称为 **reconstruction loss**，它描述的是从z这个被压缩的变量得到原来的输入x的不精确程度；而第二项被称为 **KL penalty**。这两项都相对而言容易计算。当然，为了使得计算更加简便，我们一般取如下的方式：

1. $p(z;\theta)\sim N(0,I)$（这里其实不依赖于θ）

2. $p(x|z;\theta)\sim N(f(z;\theta),I)$

3. $q(z|x;\phi)\sim N\Big(\mu(x;\phi),\mathrm{diag}\big(\exp\big(\sigma(x;\phi)\big)\big)\Big)=N\big(\mu(x;\phi),\mathrm{diag}(\Sigma(x;\phi))\big)$（这一取法也被称为 **Isomorphic Gaussian**）

事实上，这一思想就恰恰被著名的 **Variational Autoencoder** 模型所利用。



我们来回顾一下前面给出的 model 的结构：训练两个网络，分别给出$p(x,z;\theta)$和$q(z|x;\theta)$。当我们取$p(z)\sim N(0,1)$的时候，我们也就是指定了$p(x|z;\theta)$和$q(z|x;\phi)$。在训练θ的时候，我们的目标是

$$\sum_{x\in D}\sum_{z\sim q(z|x;\phi)}\log\frac{p(z;\theta)p(x|z;\theta)}{q(z|x;\phi)}$$

而在训练ϕ的时候，我们的目标是 KL divergence：

$$\sum_{x\in D}\sum_{z}q(z|x;\phi)\log\frac{p(z,x;\theta)}{q(z|x;\phi)}$$
$$=\sum_{x\in D}\left(E_{z\sim q(z|x;\phi)}\log p(x|z;\theta)-KL\big(q(z|x;\phi)||p(z;\theta)\big)\right)$$

整个模型可以被画成右图的结构：输入一个x，通过神经网络计算μ和Σ，进而生成z的分布（也就是q）；接下来，对于生成的部分，取样z并经过网络$p(x|z;\theta)$，就又给出了x。这也就是为何此网络被称为“Autoencoder”。

我们接下来考虑这个模型该如何训练：首先注意到我们取$p(z;\theta)$实际上不依赖于θ，因此我们会发现无论是训练θ还是ϕ，其有效的目标都是

$$\sum_{x\in D}\left(E_{z\sim q(z|x;\phi)}\log p(x|z;\theta)-KL\big(q(z|x;\phi)||p(z)\big)\right)$$

这也就恰是 ELBO。因此，Autoencoder 的训练目标就是**训练θ,ϕ共同优化 ELBO**。接下来，我们可以发现对于前面给出的q和p的形式，因为它们都是高斯分布，第二项 KL penalty 可以给出解析解，其对于ϕ的梯度也容易给出；但第一项 reconstruction loss 虽然很快可以给出对于θ的梯度，给出ϕ的梯度却不太容易，因为ϕ的形式隐式地出现在取样的分布里。这时，我们就要用到一个重要的思路——**re-parameterization trick** 了。注意到如果令

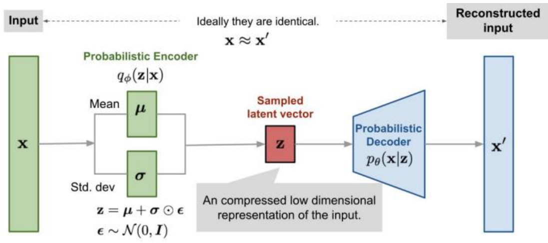
$$z=\mu(x;\phi)+\Sigma(x;\phi)*\epsilon$$

（和之前一样，$*$代表分量积）那么$\epsilon\sim N(0,1)$。这样，这个表达式就变成了

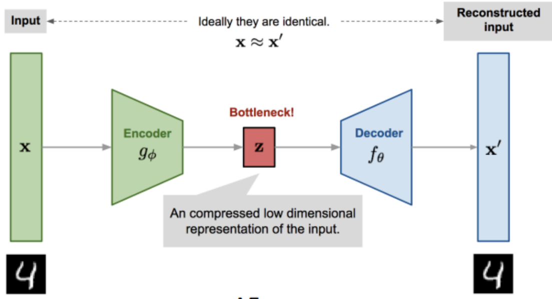
$$L(\theta,\phi)=\sum_{x\in D}\left(E_{\epsilon\sim N(0,1)}\log p(x|z=\mu(x;\phi)+\Sigma(x;\phi)*\epsilon;\theta)-KL\big(q(z|x;\phi)||p(z)\big)\right)$$

这就是 Variational Autoencoder 的最终目标。

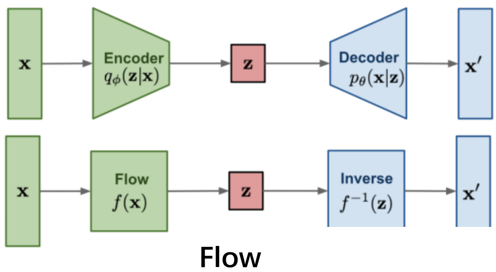
我们来把 Variational Autoencoder 和普通的 Autoencoder 以及之前的 Flow model 做一些对比：相比于普通的 Autoencoder 而言，Variational Autoencoder 的 encoder 和 decoder 网络不再是 deterministic 的，而是通过概率分布的方式来训练。一个不错的观点是，如果采用前面 re-parameterization 的思路，当我们取高斯分布的噪声$\epsilon = 0$的时候，网络就相当于 encoder，因此 Variational Autoencoder 可以大概理解为增加了噪声的 Autoencoder。



VAE

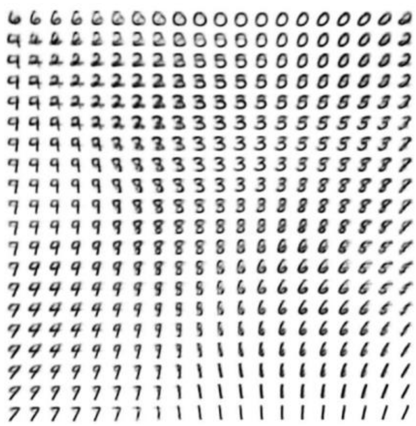


AE



Flow

而相比于 Flow model 而言，虽然二者都有一个中间的 latent variable z，但是 Flow model 要求 bijection，导致z对应的维度必须和x一致，因此不容易对高维的数据给出高效的训练过程；而 Variational Autoencoder 并没有明确要求z的维度大小，因此可以进行一定的“唯独压缩”。作为总结，VAE（也就是 Variational Autoencoder）具有 **approximation inference、dimension reduction 和 flexible architecture**。同时，对于连续的z，p可以生成连续变化的x，这使得 Autoencoder 得以“提取”出样本的一些特征，如右下图所示。



我们还可以讨论 VAE 的其他使用方式：它是否可以像 Energy-based model 一样用来 inpainting（也就是恢复被覆盖的像素）呢？我们可以试着给出这样的思路：假设x（原始图片）经过 mask 操作得到$\bar{x}$，我们希望通过

$$\bar{x} \xrightarrow{q(z|\bar{x})} z \xrightarrow{p(x|z)} x$$

得到x。这样的操作能够成功的前提是，我们必须有

$$q(z|\bar{x}) \approx q(z|x)$$

也就是说q网络必须对于输入比较 robust。那么如何能做到这一点呢？我们可以考虑在**训练的时候就随即地去掉部分 input pixel**，也就是把一些位置设成 0。事实上，这个方法在实验中得到了很好的效果。

另外一个有趣的话题是 **Conditional VAE**，也就是对于 labeled data 进行训练，在生成的时候也要指定一个 label 进行特定类型图片的生成。实现这一点并非困难，我们可以直接把原先的 encoder 和 decoder 变成

$$q(z|x, y; \phi), p(x|y, z; \theta)$$

其中y代表 label。一个更具有挑战性的任务是所谓 **Semi-supervised learning**，也就是输入一部分含有 label，另一部分没有 label 的情况。这时，一般取

$$p(x|y,z;\theta), q(z,y|x;\phi)$$

也就是对于每一个输入x都产生一个 latent variable z和 label y。实际应用中为了方便，我们一般取

$$q(z,y|x;\phi) = q(z|x;\phi)q(y|x;\phi)$$

对于这一任务，在训练的时候我们可以在 labeled data 上面训练$q(y|x;\phi)$，也就是对这个q和真实的分布进行 cross entropy loss；但是对于 unlabeled data，情况变得略微复杂。我们回顾原先的 loss function：

$$L(\theta,\phi;x) = E_{z\sim q(z,y|x;\phi)}\left[\log\left(p(x|z;\theta)\right)\right] - KL(q(z|x;\phi)||p(z))$$

现在，为了同时也训练$q(y|x;\phi)$，我们把y对应的项加进去：

$$\begin{aligned} L_{\text{new}}(\theta,\phi;x) = E_{z,y\sim q(z,y|x;\phi)}\left[\log\left(p(x|z,y;\theta)\right)\right] - KL(q(z|x;\phi)||p(z)) \\ - KL(q(y|x;\phi)||p(y)) \end{aligned}$$

其中$p(y)$可以被取为均匀的（在理想状态下，各个 label 的数据数目应该差不多）。我们观察这个 loss function，可以发现第二项和第三项是好算的；而第一项我们还是采用原先的 re-parameterization trick：

$$\begin{aligned} & E_{z,y\sim q(z,y|x;\phi)}\left[\log\left(p(x|z,y;\theta)\right)\right] \\ & \qquad = E_{y\sim q(y),\epsilon\sim N(0,1)}\left[\log\left(p(x|z=\mu(x;\phi)+\Sigma(x;\phi)*\epsilon,y;\theta)\right)\right] \end{aligned}$$

但是对于离散的y，我们没法按照这个方法处理。不过好在y作为 label，一般来说个数也不会特别多，因此可以最后写为

$$\begin{aligned} & E_{z,y\sim q(z,y|x;\phi)}\left[\log\left(p(x|z,y;\theta)\right)\right] \\ & \qquad = \sum_y q(y) E_{\epsilon\sim N(0,1)}\left[\log\left(p(x|z=\mu(x;\phi)+\Sigma(x;\phi)*\epsilon,y;\theta)\right)\right] \end{aligned}$$

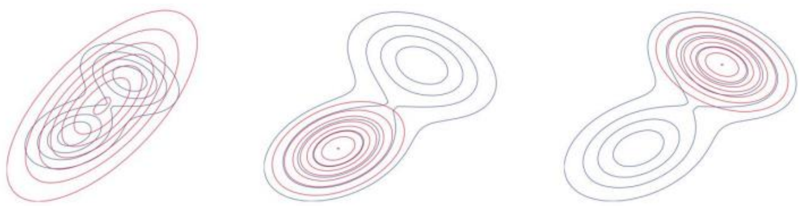
把这一项代入原先的 loss 计算，我们就完成了 Semi-supervised Learning 模型的讨论。

根据前面的讨论，我们已经可以看到 VAE 具有很多的优势，比如说相比于 Flow model 具有更加 flexible 的结构，同时 training 也比较简单。但是接下来我们主要来论述它的一些可改进空间。

首先，它的 inference 存在近似，这一本来的优势也伴随着问题的产生：对于 KL divergence，为了从好计算的q中取样，我们选取了

$$KL(q||p)=\sum_z q(z)\log\frac{q(z)}{p(z)}$$

但是相比于$KL(p||q)$而言，这一 loss 在训练q的时候会存在一些问题，也被称为 **mode collapse issue**。事实上，$KL(p||q)$称为 **forward KL**，而$KL(q||p)$称为 **reverse KL**；数学上的理论指出，对于目标p具有双峰分布的情况，forward KL 一般会产生一个大的分布覆盖两个峰（因此被称为 inclusive KL）；而 reverse KL 一般会集中在某个峰上面（因此被称为 exclusive KL）。（原因可以大致的理解为，对于 exclusive KL 而言$q(z)=0$带来的 penalty 比较小，而对于 inclusive KL 而言这个 penalty 比较大。）正是因为如此，一些人也提出使用 forward（inclusive）KL 作为 KL divergence，并做了相应的研究。



(a) (b) (c)

inclusive KL exclusive KL

第二，我们关注到在我们的模型里取定了$p(z)\sim N(0,I)$，这是为了简化计算；基于同样的原因，我们取了$q(z|x;\phi)$是高斯分布，也就是$N\big(\mu(x;\phi),\Sigma(x;\phi)\big)$。但是这有可能限制了模型的潜能——比如说，如果$q$想要模拟的$p(z|x;\theta)$是**多峰的（multi-modal）**，那么这个 proposal 就略有逊色。一些人也据此提出希望把$q(z)$换为 flow model。

第三，注意到我们在前面的估计时相当于做了 importance sampling，因此可能不够精确。针对这一问题，人们也给出了 importance-weight autoencoder，其基本思路是通过从$q(z|x;\phi)$中取更多的 sample 来得到一个更精确的对我们原始目标$\log p(x)$的近似。

最后，还是因为我们取的分布均为高斯分布，并且 loss 是 Maximum Likelihood Estimation，事实上可以发现这会导致 sample 比较模糊（blurry）。为了解决这一问题，有很多方向可以探索：改变 decoder 结构、平衡 KL penalty 和 reconstruction loss 以及干脆改变 loss function。许多重要的成果都基于对 VAE 的 loss function 的修改。比如，β −VAE 就修改整体的 loss 为

$$J(\theta,\phi;x)=E_{\epsilon\sim N(0,1)}[\log p(x|z=\mu(x;\phi)+\Sigma(x;\phi)*\epsilon;\theta)]-\beta KL\big(q(z|x;\phi)||p(z;\theta)\big)$$

其中β是 hyperparameter。注意到这一 loss 并不是乱写的，它具有一个重要的解释：它

可以理解为**最大化 reconstruction loss 的同时约束**$\boldsymbol{KL\big(q(z|x;\phi)||p(z;\theta)\big)}$**不能太大**（比如不超过$\epsilon$）**这一条件极值问题的 Lagrangian**。可以说明，$\beta = 0$的时候这个 loss 就相当于普通 Autoencoder 的 loss；而我们还可以看到$\beta = 1$的时候这个 loss 对应着 Variational Autoencoder 的 loss。不仅如此，研究发现当$\beta > 1$时，这个体系具有着把不同的 factor“抽离”到z中的效果。

不过不管如何，作为一个总结，各种 Variational Autoencoder 各自有着各自的优势；不仅如此，由于很强的随机性，可以发现实验的成功也是具有很大的 randomness 的。也因此，训练中各种 practical skills 和经验也成为了重要的部分。