VAE背后的数学原理

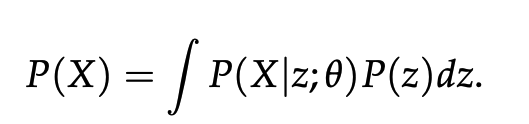
一、介绍

Generative model，学习高维数据的概率分布P(X)，学习不同维度之间的相互依赖关系，比如手写数字的生成模型如果生成了8的左边一半像素，那么剩下的像素也就随之确定了。

Latent variable【隐变量】，给与生成模型一些信息用来生成数据，

二、VAE

对于数据集中的每个样本x和他所对应的所有z，我们要做的是最大化下面的概率：



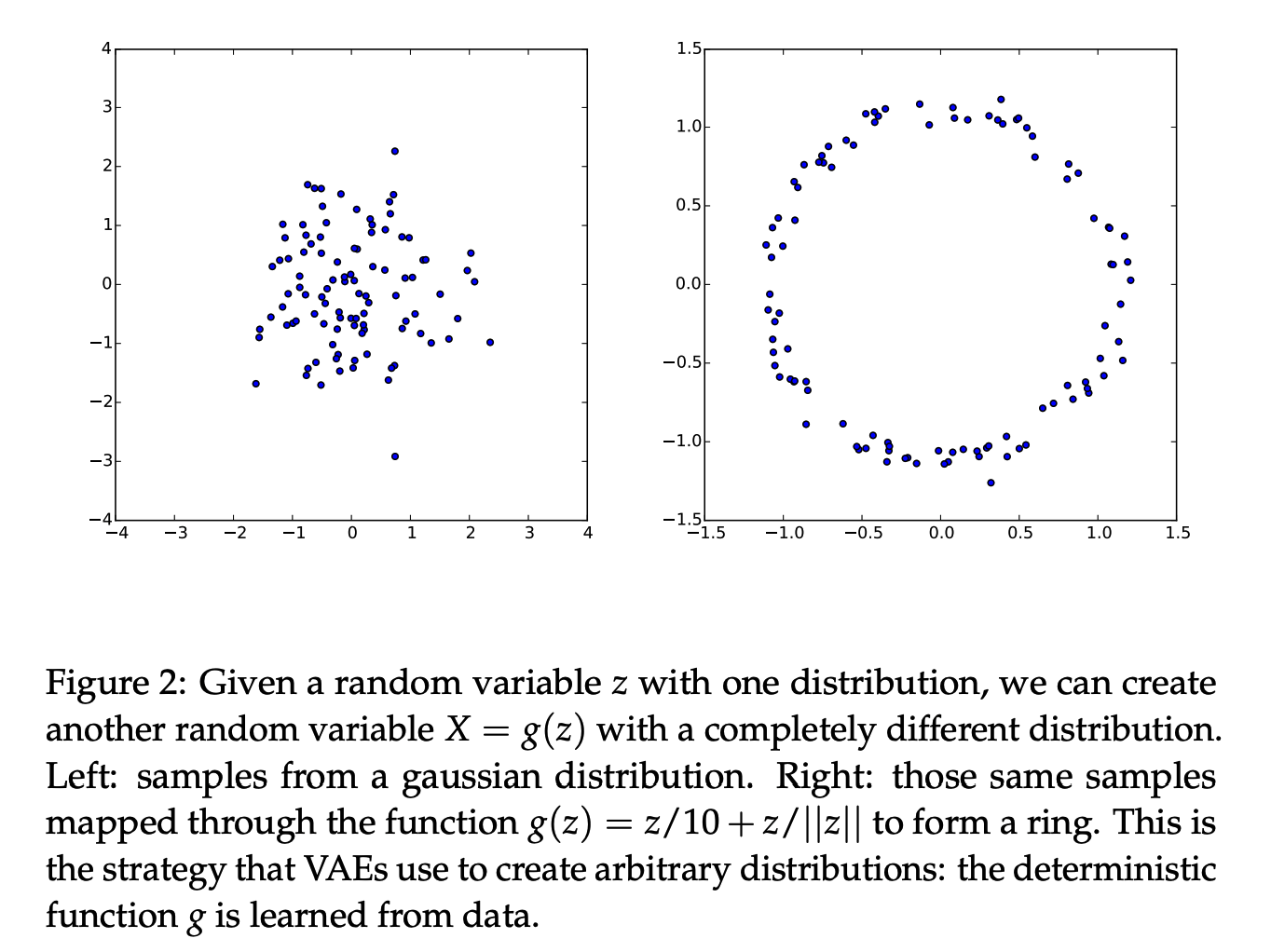
一般认为，即认为x是一个高斯分布，均值和方差都是由z经过两个对应的神经网络计算而来。

要解决上图中的公式，有两个问题需要解决：

* 1）怎样生成隐变量z
* 2）对于隐变量z要怎么处理

下面解决第一个问题：

* VAE假设z的各个维度之间是没有任何关联的，也就是相互独立。从而对于z的所有维度，都可以分别从一个一维高斯上抽样，然后concat到一起。之所以这样认为，是因为理论上z经过一个足够复杂的变换，可以变成任意一种分布，比如下图1中的例子所示，所以从这个角度来说，与其说z是隐变量，不如说z变换后的向量是隐变量，即z ->隐变量 ->x。



下面是对于第二个问题（对于隐变量z怎么处理）的处理：

单纯的想通过采样来计算P(X) = 是不可以的，因为样本数无法做到足够大。事实上，对于绝大多数的z来说，生成的X都是不存在的，即的值几乎为0；这对于计算这些求和是没什么用的。因此，我们需要想办法只抽那些能生成X的z，即减小z的采样空间；所以我们前面说的P()也就变成了P(z|X)（省略了下标i; 这里P(z|X)仍然是一个高斯分布），同时我们还需要另外一个分布Q(z|X)来逼近P(z|X)，从而生成我们想要的z。

接下来引入一个概念叫做KL散度(KL divergence)，用来衡量两个分布之间的差别：

比如，我们希望衡量Q(x)和P(x)两个分布之间的差别，有如下的计算方法：

所以可得：

【贝叶斯变换可以得到上面这个公式】

对于上式进行移项可得：

上式的左边包括了我们想要maximize的logP(X)，还包含了一个误差项D[Q(z|X)||P(z|X)]，其中P(X)和P(z|X)都是没办法直接计算的。我们想要maximize等式左边的东西，就等价于增大logP(X)并且减小误差，并且等式左边的第二项D(Q(z|X)||P(z|X))是让Q(z|X)去匹配P(z|X)，如果Q是一个容量密度很高的分布，Q(z|X)是能够完全覆盖P(z|X)的，那么KL散度D(Q(z|X)||P(z|X))一定是0。值的注意的是，等式右边的是一个类似于AutoEncoder的东西，其中Q(z|X)是encoder，可以理解为把一张图片变成一个随机数，并且让这个随机数接近一个正态分布P(z)，从而方便采样；P(X|z)是Decoder，把采样得到的随机数映射为图片。

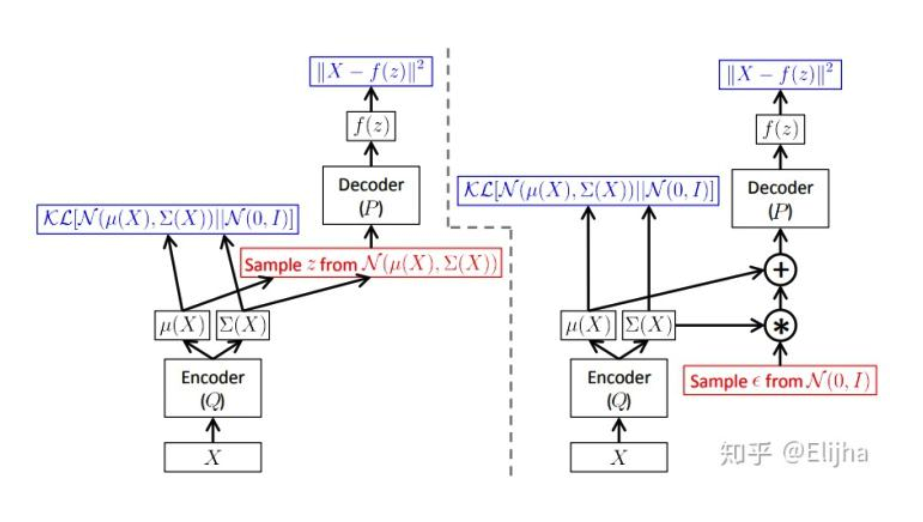
下面是对优化目标函数的介绍：

现在我们要优化的目标就是上面的等式右边的值。

首先我们需要确定概率密度的形式；一般采用正态分布，即N()，因此，第二项可以改为D(N() ||N(0,I))，其中分别用两个神经网络来完成。接下来，我们就可以使用两个正态分布的KL散度来计算这一项了。

然后对于第一项,它不仅仅要依赖于P的参数，还需要依赖Q的参数，而上式并没有体现这种依赖，这个我们留到后面解决。

由于是采用随机梯度下降的方法来进行优化，因此对于一个batch，可以将所有单个样本的logP(X|z)求和并取平均数来作为期望的估计。但是这样出了一个问题，那就是把Q(z|X)弄丢了【也就是上面说的依赖关系没有体现】，换一个角度也可以说，每次训练的时候，梯度传不进Q里。我们需要让encoder产生能被decoder正确解码的隐向量，或者说让encoder和decoder之间是可以交互的，因此这里引入了一个trick，叫做“reparameterization trick”，如图所示：



上图中，红色框【sample操作】是一个不可back prop的操作。

左边是没有采用的reparameterization trick的图，它的sample操作在整个bp路径的中间，因此无法反传梯度；

右边的图是采用了reparameterization trick的图，用encoder计算出和(方差，即)之后，不是直接在这个分布上采样，而是在另外的一个标准正态分布上采样之后，乘以再加上，通过这两种方法，就可以把不可bp的操作转移到了梯度回传的路径外面。

至此，整个VAE就可以训练了。