学习基础知识的笔记

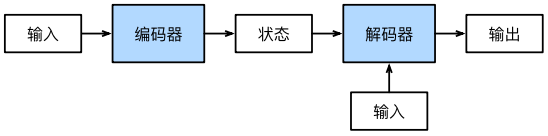
~~虽说是基础知识，但是对我一个没有概率论基础的人来说确实有点超纲了。。。~~
好在还是学完了。最难的部分莫过于数学。
概率论的有些基本概念会没有太掌握，但是整个VAE数学推理的东西总算是学完了。
~~但是又好像什么都没懂。。。~~
本笔记主要包括抽象的概念理解、理论推导，而非代码实现。我对Latex语法不熟，有一些数学公式实在是不好打，有一部分公式的推理、我对它的一些理解是记在纸上的。望佬见谅。

Encoder and Decoder:

编码器-解码器更像是一种思维，而非一个全新的架构。

机器翻译是序列转换模型的一个核心问题， 其输入和输出都是长度可变的序列。

为了处理这种类型的输入和输出，
我们可以设计一个包含两个主要组件的架构：
第一个组件是一个*编码器*（encoder）：
它接受一个长度可变的序列作为输入，
并将其转换为具有固定形状的编码状态。
第二个组件是*解码器*（decoder）：
它将固定形状的编码状态映射到长度可变的序列。
这被称为*编码器-解码器*（encoder-decoder）架构，
如 :numref: `structure of encoder-decoder` 所示。



🏷 `structure of encoder-decoder`

我们以英语到法语的机器翻译为例：
给定一个英文的输入序列：“They”“are”“watching”“.”。
首先，这种“编码器 - 解码器”架构将长度可变的输入序列编码成一个“状态”，
然后对该状态进行解码，
一个词元接着一个词元地生成翻译后的序列作为输出：
“Ils”“regordent”“.”。

KL 散度（KL divergence）

> 相对熵（relative entropy）又称为 KL 散度（Kullback–Leibler divergence，简称 KLD），信息散度（information divergence），信息增益（information gain）。
>
> KL 散度是两个概率分布 P 和 Q 差别的非对称性的度量，用来度量使用基于 Q 的编码来编码来自 P 的样本平均所需的额外的位元数。 典型情况下，P 表示数据的真实分布，Q 表示数据的理论分布，模型分布，或 P 的近似分布。

【定义】

对于离散随机变量，其概率分布 P 和 Q 的 KL 散度可按下式定义为：

$$D_{KL}(P||Q) = -\sum_i P(i) ln \frac{Q(i)}{P(i)}$$

等价于

$$D_{KL}(P||Q) = \sum_i P(i) ln \frac{P(i)}{Q(i)}$$

即按概率 P 求得的 P 和 Q 的对数商的期望值。KL 散度仅当概率 P 和 Q 各自总和均为 1，且对于任何 i皆满足 $P(i) > 0$及 $Q(i) > 0$时，才有定义。

对于连续随机变量，其概率分布 P 和 Q 可按积分方式定义为：

$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) ln \frac{p(x)}{q(x)} dx$

其中 p和 q分别表示分布 P和 Q的密度。

【特性】

1. 相对熵的值为非负数：

$D_{KL}(P||Q) \geq 0$

由吉布斯不等式可知，当且仅当 $P = Q$时 $D_{KL}(P||Q)$为零。

1. 尽管从直觉上 KL 散度是个度量或距离函数, 但是它实际上并不是一个真正的度量或距离（not a distance metric）。因为 KL 散度不具有对称性：从分布 P 到 Q 的距离通常并不等于从 Q 到 P 的距离。

$D_{KL}(P||Q) \neq D_{KL}(Q||P)$

重参数化

重参数化核心原理比较简单，而实现又比较繁琐。有很多种的方法很多佬都已经总结得很清楚，这里不再赘述。

重参数化的出现是为了解决这样一个问题：我们用这个更复杂的模型，就精度更高，但是预测得更慢，用户不喜欢；用更简单的模型，就预测得更快，但精度更低，用户也不喜欢。有没有一种办法，让我们可以精度慢，同时用得快？

在这个诉求下，重参数化横空出世。

重参数化的创始人之一丁霄汉博士认为它的基本思想是：我们把模型分成架构+参数。我们完全可以把它们俩分开看。

但我觉得这个基本思想不够精确。参数完全就是跟着架构走的嘛！只不过，我们用了一套架构+一套参数来训练，再把它变成另一套架构+另一套参数。

我觉得创立重参数化很妙的一点就是：由于卷积的线性性，明明$3*3+1*3+3*1$卷积和$3*3$卷积可以认为等价，但是！用$3*3+1*3+3*1$来train然后再叠加起来，就是好于直接用$3*3$train!

可是直觉上来看，它们明明是等价的呀？为什么训出来就是不一样呢？丁霄汉博士给出的解释是：由于模型动态变化的复杂性。前向传播计算等价不等于反向传播梯度下降等价。

科学上也有很多反直觉的东西：相对论“距离、质量、时间和速度有关”；数学上也有：哥德尔不完备性定理、黎曼几何中的“假定两平面中有两条永不相交的不平行的直线”。这样的理论有些人一看就认为是歪门邪道，可是实际上它们非常有用。可能这种敢于“反直觉”就是科学家们得以发现新东西的原因。

学习是对直觉的修正。

————杨振宁

自编码器

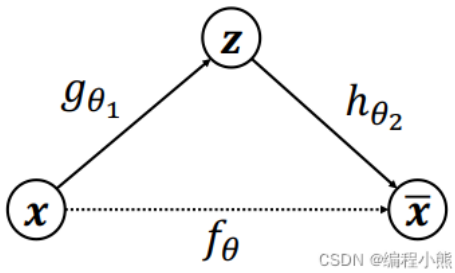
自编码器原理

自编码器算法属于自监督学习范畴，如果算法把 x 作为监督信号来学习，这里算法称为自监督学习 (Self-supervised Learning)

在监督学习中神经网络的功能:
$o = f(x), x : R^{d_{in}}, o : R^{d_{out}}$
。
d_{in}
是输入的特征向量长度，
d_{out}
是网络输出的向量长度。对于分类问题，网络模型通过把长度为
d_{in}
输入特征向量$\boldsymbol{x}$变换到长度为
d_{out}
的输出向量$\boldsymbol{o}$，这个过程可以看成是特征降维的过程，把原始的高维输入向量$\boldsymbol{x}$变换到低维的变量$\boldsymbol{o}$。

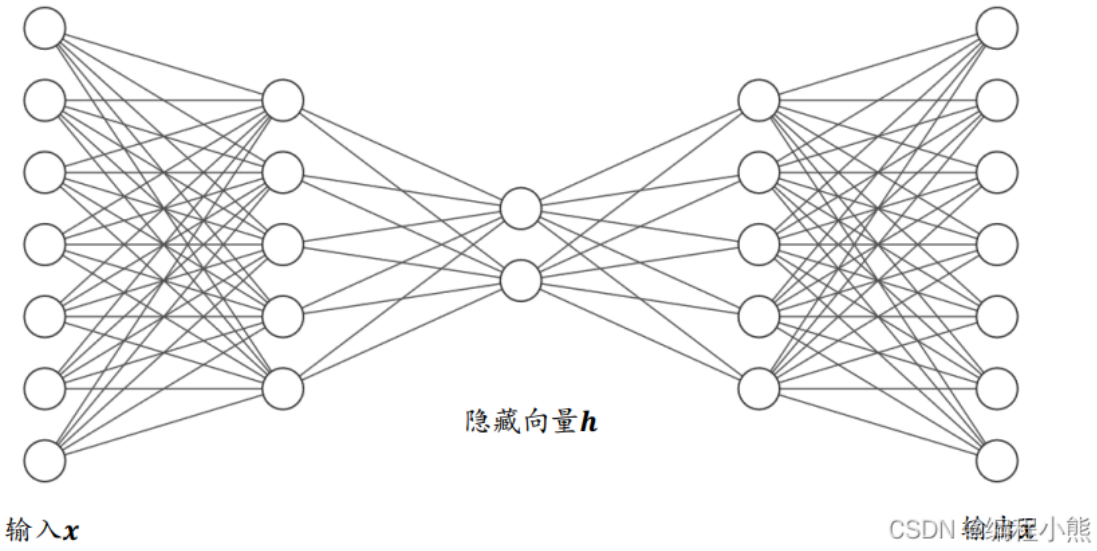
特征降维 (Dimensionality Reduction)在机器学习中有广泛的应用， 比如文件压缩 (Compression)、 数据预处理(Preprocessing) 等。 最常见的降维算法有主成分分析法 (Principal components analysis， 简称 PCA)，通过对协方差矩阵进行特征分解而得到数据的主要成分，但是 PCA 本质上是一种线性变换，提取特征的能力极为有限

利用神经网络的强大非线性表达能力去学习到低维的数据表示，但是训练神经网络一般需要一个显式的标签数据 (或监督信号)，但是无监督的数据没有额外的标注信息，只有数据$\boldsymbol{x}$本身
利用数据$\boldsymbol{x}$本身作为监督信号来指导神经网络的训练，即希望神经网络能够学习到映射
f_θ
: $\boldsymbol{x} \rightarrow \boldsymbol{x}$。



把网络
f_θ
切分为两个部分，前面的子网络尝试学习映射关系:
$g_{\theta 1}$
: $\boldsymbol{x} \rightarrow \boldsymbol{z}$，后面的子网络尝试学习映射关系
$h_{\theta 2}$
: $\boldsymbol{z} \rightarrow \boldsymbol{x}$。 把
$g_{\theta 1}$
看成一个数据编码 (Encode) 的过程，把高维度的输入$\boldsymbol{x}$编码成低维度的隐变量$\boldsymbol{z}$(Latent Variable，或隐藏变量)，称为 Encoder 网络 (编码器);
$h_{\theta 2}$
看成数据解码 (Decode) 的过程，把编码过后的输入$\boldsymbol{z}$解码为高维度的$\boldsymbol{x}$， 称为 Decoder 网络 (解码器)

编码器和解码器共同完成了输入数据$\boldsymbol{x}$的编码和解码过程，把整个网络模型
f_θ
叫做自动编码器 (Auto-Encoder)，简称自编码器。 如果使用深层神经网络来参数化
$g_{\theta 1}$
和
$h_{\theta 2}$
函数， 则称为深度自编码器 (Deep Auto-encoder)



自编码器能够将输入变换到隐藏向量$\boldsymbol{z}$，并通过解码器重建 (Reconstruct，或恢复) 出$\boldsymbol{x}$。解码器的输出能够完美地或者近似恢复出原来的输入，即$\boldsymbol{x} \approx \boldsymbol{x}$,
自编码器的优化目标:
$Minimize\tau = dist(x, \bar{x}), \bar{x} = h_{\theta 2}(g_{\theta 1}(x))$
, dist($\boldsymbol{x}$, $\boldsymbol{x}$) 表示 $\boldsymbol{x}$和
$\bar{x}$
的距离度量，称为重建误差函数。 常见的度量方法有欧氏距离的平方，计算方法如下：$\mathcal{L}$ = ∑(xi - $\bar{x}i$)2
它和均方误差原理上是等价的。 自编码器网络和普通的神经网络并没有本质的区别， 只是训练的监督信号由标签$\boldsymbol{y}$变成了自身$\boldsymbol{x}$。 借助于深层神经网络的非线性特征提取能力， 自编码器可以获得良好的数据表示，相对于 PCA 等线性方法，自编码器性能更加优秀， 甚至可以更加完美的恢复出输入$\boldsymbol{x}$

在encoder-decoder中，若我们用的是普通的AE，而不是VAE，直观上有两个个很明显的问题：

1. 假如我们在潜变量空间中任意取一个点，丢到decoder里面，它就给我们一个很奇怪的结果。这对应李宏毅老师的：人看不出来它想画的东西是宝可梦。

2. 如果在和训练数据A、B对应的潜在变量之间选择一个中间点，再decode，人是看不出来它以何种姿态“介于A和B之间”的。这就对应李宏毅老师“半月”和“全月”的例子。

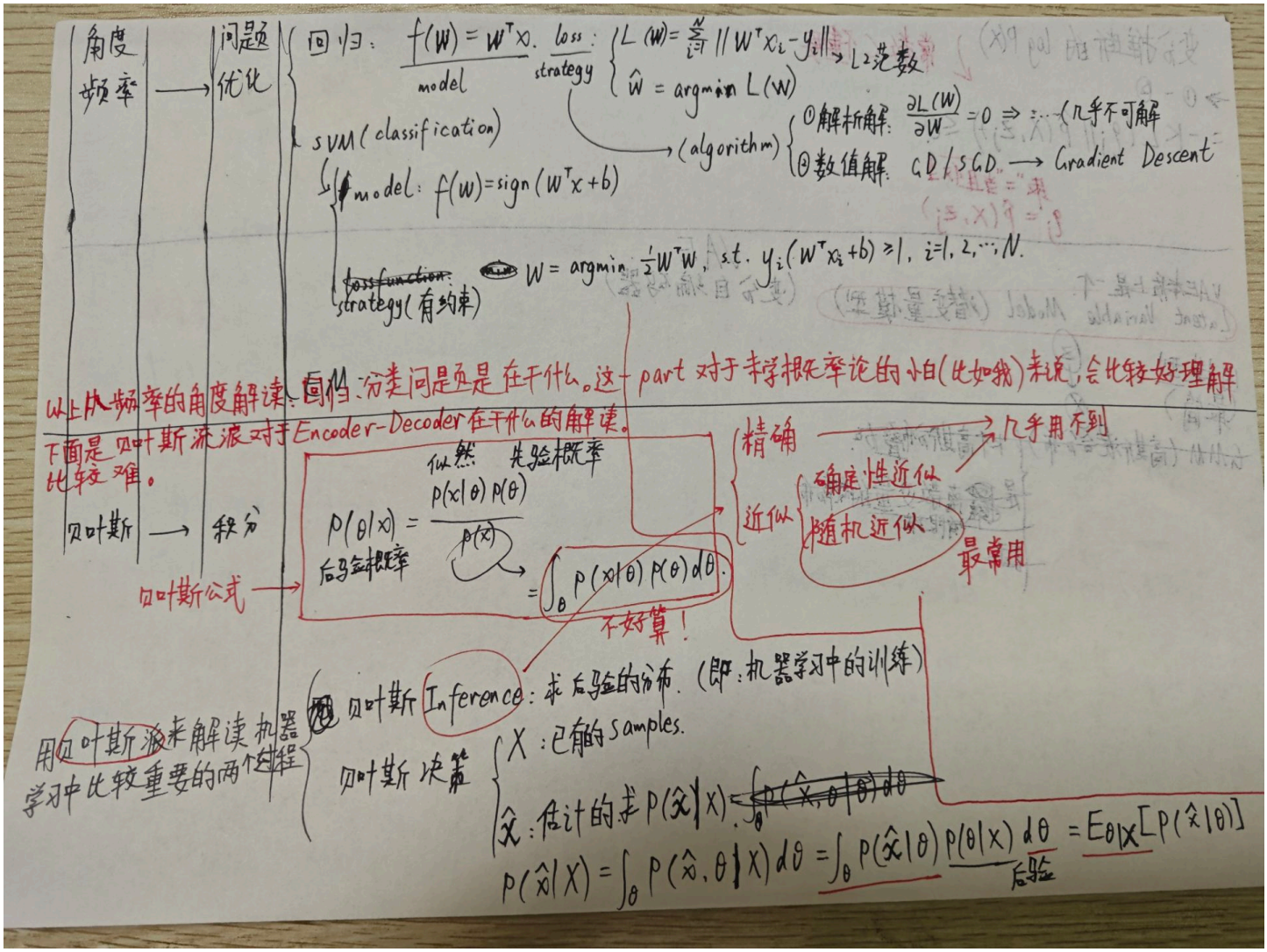
可以认为：如果我们想要在潜变量空间中任取一个点，就生成“看起来像一回事”的结果，和VAE相比，AE是一个很挫的模型。因为它本质上就是一个多层感知机。这就引出了这道题的hero：VAE

由于VAE就是用来实现变分推理的，下面一起讲变分推理&VAE。

变分推理

首先，我们分别从频率、贝叶斯流派的角度，来看一下：机器学习是在干什么？

下图中一些字母的意思：
X:已有的N个样本
$\hat{x}$:新的一个样本，是需要我们的模型推断的
$P(\hat{x}|X)$:已经学习了N个样本的情况下，我们的模型预测$\hat{x}$的概率。
θ:参数。



很显然，我们最大的工作量应该在于贝叶斯推断，而贝叶斯推断本质上是求后验概率的分布。我写出来的贝叶斯公式中，什么最不好求？是分母$P(X)$！为什么呢。它是一个实际工程中几乎积不出来的积分。为了近似这个分母，我们就要用到变分推理。

下面讲变分推理

简单来讲，变分推理是用可控概率分布来近似复杂概率分布的技术。它尝试通过一个更简单、可控的概率分布来逼近目标分布，以便于计算和分析。变分推理的关键在于选择一个合适的简单分布，然后调整这个分布的参数，让它尽可能接近真实的复杂分布。

举一个简单的例子：我有一个很复杂的概率分布。我想用很多个高斯（正态）分布叠加起来，来近似这个分布（这就是大名鼎鼎的高斯混合模型：GMM）。数学上可以证明，当加的高斯分布的个数足够多时，误差就可以降到很低很低。

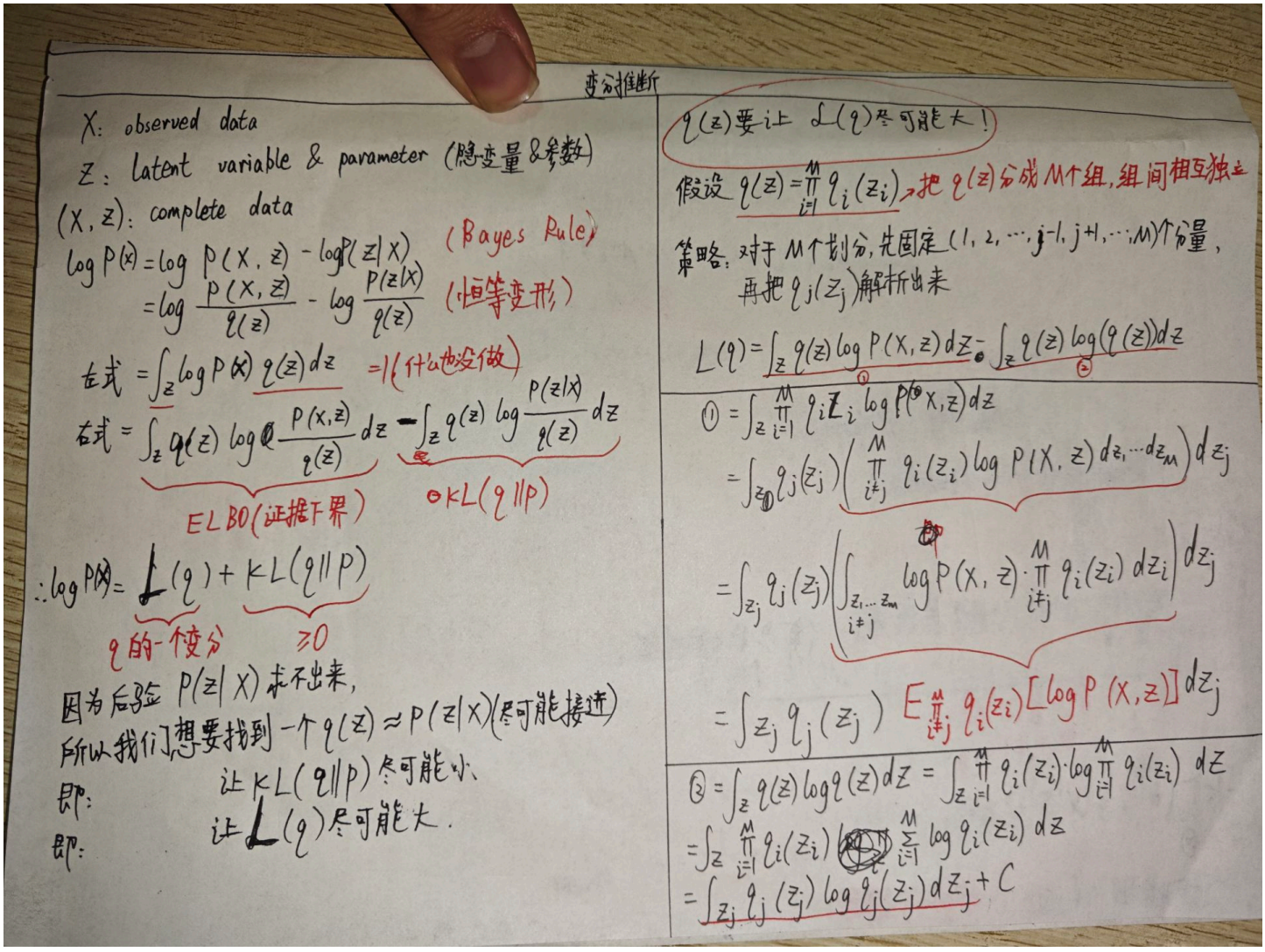
具体怎么做呢？
我们把这些高斯分布用一个离散型随机变量z标号。$z = 1, 2, ..., n$。每一个z出现的概率是$p(i)$

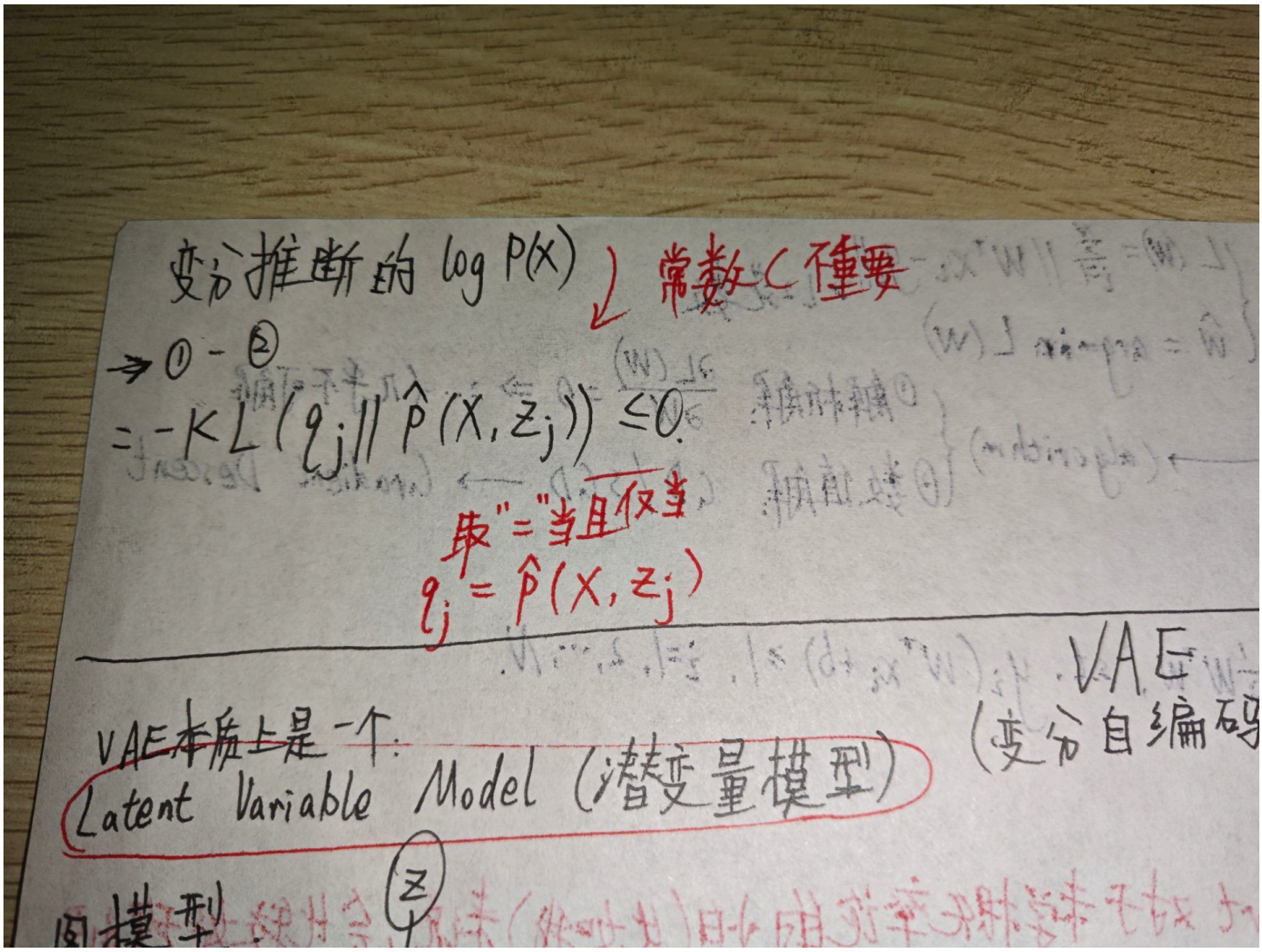
假设对于第i个高斯分布，有：$X|_{z=i} \sim N(\mu(i), \sigma(i)^2)$

问题在于：第i个高斯分布的概率$p(i)$（当然，若z是连续型随机变量，就没有这个烦恼）、均值$\mu(i)$、方差$\sigma(i)$分别是多少？答案肯定存在，但是我们是不好求出来（*确定性近似*）的，一般都要估计这很多个高斯分布的参数，这个估计的过程属于变分推理。直观上，这个误差可以用KL散度来定量衡量。

但是我们为什么可以肯定的说，要让差别最小，就必须让KL散度最小？给出$P(X)$和它的定量关系？我们真的要用这种方法吗？加入要用，我们应该盯住KL散度，还是别的什么东西？这些东西就需要严格的数学推理了。这一部分比较难，实际工程中可能应用比较少，却是一个优秀的数据科学家必备的知识。数学推理见我写的纸质笔记。

对下图的一些解释：
z:隐变量。这里包含了隐变量、参数
$P(z|X)$：理想的z关于X的概率分布。
$q(z|X)$：我们随便捡的一个z关于X的概率分布。数学上等于$q(z)$，我们想让它与理想的$P(z|X)$尽可能接近。
下图的Z全部看成小写的z。





总之定量关系就是：

$logP(X) = ELBO + KL(q(z \mid X) \ || \ P(z \mid X))$

没错，假如我们的目的仅仅是拟合$P(X)$，那就是要KL散度最小。~~（但是好像什么都没做）~~

没关系，看到VAE的损失函数就可以知道我们这里数学推理的意义了。

而用GMM来变分推理，有两个特点：

1. 隐变量Z只有一个
2. 这个Z只能取到离散的值，尽管它是一个概率分布，它也是一个离散概率分布。

对这两个特点进行改进，我们弄出来了VAE的架构。下面就要讲VAE了。

变分自编码器

前面讲了这么多，都是为了引出本题的主角：VAE（*Variational Autoencoder*）。Autoencoder：自编码器，一种架构；Variational：变分，就是说我们可以用变分推理的方法来逼近这个概率分布$P(X)$。

VAE的出现，可以看成对两个已有的东西的改进：自编码器AE和高斯混合模型GMM。

线索1：在生成式机器学习层面，对AE的改进

在听课的开始，我有个疑惑：为什么VAE要最大化$P(X)$？其实很简单：我们想要给decoder任意扔一个潜变量空间中的点，decoder预测出来的东西（图片？文字？）尽可能靠谱，尽可能像那么回事；而不是想AE那样，好像要画什么东西，但是画得人无法看懂。

从概率论上看，为了达到这个目的，我们要干2件事情：

1. 最小化$KL(q||P)$
2. 最大化ELBO。

从机器学习上看，我们要干1件事：

1. 对输入的数据加噪声，让预测的数据更鲁棒。（这是一个很挫的解释，我自己都觉得不好）

线索2：在变分推理层面，对GMM的改进

GMM理论上可以拟合一切单变量连续概率分布，但是它的两个特点，决定了加入在VA中用它（相当于code只有1个标量，而且这个标量还只能取离散的值，想想都觉得很挫。而且现在的计算机估计在梯度下降的过程中直接就爆了），对于更加复杂的概率分布模型（要画哪个图片更靠谱？）没有强大的拟合能力。同时，我们让KL散度最小、ELBO最大的具体方法，是什么呢？为了解决这些问题，我们要干3件事：

1. 把隐变量Z变成很多个隐变量，即为：一个n维向量。这个就是我们AE中的code
2. 把向量的每个分量变成相互独立的、服从连续概率分布的随机变量。在VAE里面，这个分布用正态分布。（但是数学上，不一定要选正态分布，只要是连续概率分布都可以。但是工程中为了简单，都用正态分布。）
3. 用神经网络去拟合变分推理中的函数，从而让KL散度尽可能小、ELBO尽可能大。

上面在线索1中讲到，我们要干2件事情：

1. 最小化$KL(q||P)$
2. 最大化ELBO。

ELBO变形之后，可以写成两个形式：

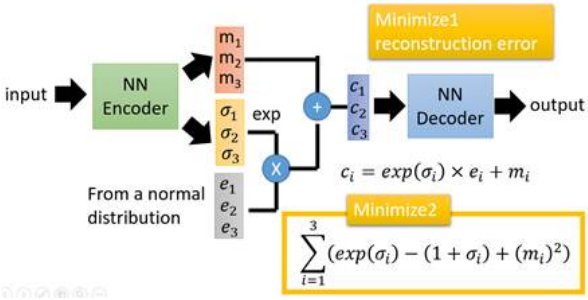
$$ELBO = \int zq(z \mid x)log(q(z \mid x)P(x \mid z)P(z))dz \quad (1)$$

$$ELBO = -KL(q(z \mid x) \ || \ P(z)) + Eq(z \mid x)log(P(x \mid z)) \quad (2)$$

这两个式子变号，再变形，就是李宏毅老师讲的工程中要应用的损失函数的由来。

$$\sum_{i=1}^{J}(exp(\sigma_i) - (1+\sigma_i) + (m_i)^2)$$

综合以上两条线索，我们就弄出来了VAE的模型架构：



上面这张图就是 VAE 的模型架构，我们先粗略地领会一下这个模型的设计思想。

在 auto-encoder 中，编码器是直接产生一个编码的，但是在 VAE 中，为了给编码添加合适的噪音，编码器会输出两个编码，一个是原有编码 (m_1,m_2,m_3)，另外一个是控制噪音干扰程度的编码 ($\sigma_1,\sigma_2,\sigma_3$)，第二个编码其实很好理解，就是为随机噪音码 (e_1,e_2,e_3) 分配权重，然后加上 exp(σ_i) 的目的是为了保证这个分配的权重是个正值，最后将原编码与噪音编码相加，就得到了 VAE 在 code 层的输出结果 (c_1,c_2,c_3)。其它网络架构都与 Deep Auto-encoder 无异。

损失函数方面，除了必要的重构损失外，VAE 还增添了一个损失函数（见上图 Minimize2 内容），这同样是必要的部分，因为如果不加的话，整个模型就会出现问题：为了保证生成图片的质量越高，编码器肯定希望噪音对自身生成图片的干扰越小，于是分配给噪音的权重越小，这样只需要将 ($\sigma_1,\sigma_2,\sigma_3$) 赋为接近负无穷大的值就好了。所以，第二个损失函数就有限制编码器走这样极端路径的作用，这也从直观上就能看出来，exp(σ_i)-(1+ σ_i) 在 σ_i =0 处取得最小值，于是 ($\sigma_1,\sigma_2,\sigma_3$) 就会避免被赋值为负无穷大。

但是，你可能会问：为什么这个损失函数必须长这个样子呢？为什么不用$(x-1)^2$这样的函数？

这个其实就是由(1)或者(2)中任意一个得来的了。我们想要的是让ELBO尽可能大，也就是让-ELBO尽可能小。最后就可以变形得到这个式子。

总结：

可能讲的不是很清楚，但实在是尽力了。谢谢观看！