# HOMEWORK 4: Deep Generative Model

**Machine Learning and Data Mining (Fall 2023)**

**Student Name: 刘俊杰**

**Student ID: 21307174**

**Lectured by: Shangsong Liang**

**Sun Yat-sen University**

Your assignment should be submitted to the email that will be provided by the TA

Deadline of your submission is: 23:59PM, January 05, 2024

**Do NOT Distributed This Document and the Associated Datasets**

## Problem: Implementing the Variational Autoencoder (VAE)

For this problem, we will be using PyTorch to implement the variational autoencoder (VAE) and learn a probabilistic model of the MNIST dataset of handwritten digits. Formally, we observe a sequence of binary pixels $x \in \{0,1\}^d$, and let $z \in \mathbb{R}^k$ denote a set of latent variables. Our goal is to learn a latent variable model $p_\theta(x)$ of the high-dimensional data distribution $p_{\text{data}}(x)$.

The VAE is a latent variable model that learns a specific parameterization $p_\theta(x) = \int p_\theta(x,z)dz = \int p(z)p_\theta(x \mid z)dz$. Specifically, the VAE is defined by the following generative process:

$$p(z) = \mathcal{N}(z \mid 0, I)$$
$$p_\theta(x \mid z) = \text{Bernoulli}\left(x \mid f_\theta(z)\right)$$

In other words, we assume that the latent variables $z$ are sampled from a unit Gaussian distribution $\mathcal{N}(z \mid 0, I)$. The latent $z$ are then passed

through a neural network decoder $f_\theta(\cdot)$ to obtain the parameters of the $d$ Bernoulli random variables which model the pixels in each image.

Although we would like to maximize the marginal likelihood $p_\theta(x)$, computation of $p_\theta(x) = \int p(z)p_\theta(x \mid z)dz$ is generally intractable as it involves integration over all possible values of $z$. Therefore, we posit a variational approximation to the true posterior and perform amortized inference as we have seen in class:

$$q_\phi(z \mid x) = \mathcal{N}\left(z \mid \mu_\phi(x), \operatorname{diag}\left(\sigma^2_\phi(x)\right)\right)$$

Specifically, we pass each image $x$ through a neural network which outputs the mean $\mu_\phi$ and diagonal covariance $\operatorname{diag}\left(\sigma^2_\phi(x)\right)$ of the multivariate Gaussian distribution that approximates the distribution over latent variables $z$ given $x$. We then maximize the lower bound to the marginal log-likelihood to obtain an expression known as the evidence lower bound (ELBO):

$$\log p_\theta(x) \geq \operatorname{ELBO}(x; \theta; \phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{D}_{KL}\left(q_\phi(z \mid x) \| p(z)\right)$$

Notice that the ELBO, as shown on the right-hand side of the above expression, decomposes into two terms: (1) the reconstruction loss: $-\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x \mid z)\right]$, and (2) the Kullback-Leibler (KL) term: $\mathrm{D}_{KL}\left(q_\phi(z \mid x) \| p(z)\right)$.

Your objective is to implement the variational autoencoder by modifying utils.py and vae.py.

# 实验内容

## 1. Implement the reparameterization trick in the function sample_gaussian of utils .py. Specifically, your answer will take in the mean $\mu$ and variance $\sigma^2$ of the Gaussian

## distribution $q_\phi(z \mid x)$ and return a sample $z \sim q_\phi(z \mid x)$.

在深度学习中，通常使用梯度下降等优化算法来最小化损失函数，而这些算法要求函数是可导的。

然而，对于含有随机性的操作，比如从概率分布中采样，其梯度通常是不可导的。为了解决这个问题，引入了重参数化技巧。

在高斯分布的采样中，传统的方法是直接从给定的均值 $\mu$ 和方差 $\sigma^2$ 的正态分布中进行采样。然而，这样的采样操作是不可导的，因为它包含了随机性。

为了使采样过程可导，我们引入一个额外的随机性，并通过对参数的确定性变换来间接实现采样。具体而言，在高斯分布中，我们可以使用标准正态分布的采样和线性变换来得到所需的样本。

对于给定的均值 $\mu$ 和方差 $\sigma^2$，可以通过以下步骤来实现重参数化：

1. 从标准正态分布 $N(0, 1)$ 中采样得到一个随机数 $\epsilon$。
2. 使用线性变换 $z = \mu + \sqrt{\sigma^2} \cdot \epsilon$ 得到最终的样本 $z$。

这里，$\epsilon$ 是从标准正态分布中采样的随机数，通过这个随机数和确定性的线性变换，我们实现了对高斯分布的采样，并且整个过程是可导的，因此适用于深度学习中的梯度计算。在代码中，就是通过 `z = m + torch.sqrt(v) * epsilon` 这一步骤来实现的。

代码实现

```
def sample_gaussian(m, v):
    """
    Element-wise application reparameterization trick to sa

    Args:
        m: tensor: (batch, ...): Mean
        v: tensor: (batch, ...): Variance

    Return:
        z: tensor: (batch, ...): Samples
    """

################################################################
    # 待办：在此处修改/完成代码
    # Sample z

################################################################
    # 从标准正态分布中生成随机样本
    epsilon = torch.randn_like(m)
    # 重参数化
    z = m + torch.sqrt(v) * epsilon

################################################################
    # 代码修改结束

################################################################
    return z
```

2. Next, implement negative ELBO bound in the file vae.py. Several of the functions in utils . py will be helpful, so please check what is provided. Note that we ask for the negative ELBO, as PyTorch optimizers minimize the loss function. Additionally, since we are computing the negative ELBO over a mini-batch of data $\left\{x^{(i)}\right\}_{i=1}^{n}$, make sure to

compute the average $-\frac{1}{n}\sum_{i=1}^{n}\text{ELBO}\left(x^{(i)};\theta;\phi\right)$ over the minibatch. Finally, note that the ELBO itself cannot be computed exactly since exact computation of the reconstruction term is intractable. Instead, we ask that you estimate the reconstruction term via Monte Carlo sampling:

$$-\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] \approx -\log p_\theta\left(x \mid z^{(1)}\right)$$

here $z^{(1)} \sim q_\phi(z \mid x)$ denotes a single sample. The function kl_normal in utils.py will be helpful. Note: negative ELBO bound also expects you to return the average reconstruction loss and KL divergence.

**原理解释：**

VAE通过最大化证据下界从潜在变量中生成与输入数据相似的样本

证据下界（ELBO）是由KL散度和重构误差组成的。KL散度衡量了潜在变量分布与先验分布之间的差异，而重构误差则测量了模型生成的样本与输入数据的相似程度。由于KL散度和重构误差都是非负的，最大化ELBO等价于最小化负ELBO。

具体地，对于一个观测数据（输入数据）$x$通过编码器（Encoder）将其映射到潜在空间中的分布$q_\phi(z \mid x)$，其中$\phi$是编码器的参数。然后，我们使用重参数化技巧从这个分布中采样一个潜在变量$z$。

接下来，使用解码器（Decoder）将潜在变量$z$映射回输入空间，得到生成的数据$x'$。希望生成的数据$x'$与原始输入数据$x$相似。

重构误差即为$\log p_\theta(x \mid z)$，其中$\theta$是解码器的参数。由于这个项的精确计算是困难的，我们使用Monte Carlo采样的方式进行估计，即$-\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] \approx -\log p_\theta(x \mid z^{(1)})$，其中$z^{(1)}$是从潜在分布中采样的一个样本。

KL散度项衡量了潜在变量分布$q_\phi(z \mid x)$与先验分布$p(z)$之间的差异。KL散度的计算可以使用utils.py中提供的`kl_normal`函数。

最终，负ELBO等于KL散度和重构误差之和。目标是最小化这个负ELBO，从而使生成的数据更好地匹配输入数据。

## 代码实现

```
    def negative_elbo_bound(self, x):
        """
        Computes the Evidence Lower Bound, KL and, Reconstru

        Args:
            x: tensor: (batch, dim): Observations

        Returns:
            nelbo: tensor: (): Negative evidence lower boun
            kl: tensor: (): ELBO KL divergence to prior
            rec: tensor: (): ELBO Reconstruction term
        """

#################################################################
        # 待办：在这里修改/完善代码
        # 计算负的证据下界（ELBO）及其 KL 分解和重构（Rec）分解
        #
        # 注意 nelbo = kl + rec
        #
        # 输出结果应该都是标量

#################################################################
       # 对输入数据 x 进行编码，得到编码的均值 m 和方差 v
        m, v = self.enc.encode(x)
        # 使用重参数化技巧从编码中采样得到潜在变量 z
        z = ut.sample_gaussian(m, v)
        # 使用解码器将潜在变量 z 解码为生成的数据 z
        z = self.dec.decode(z)
        # 计算重构误差，即生成数据与输入数据的负对数伯努利交叉熵
        rec = torch.mean(-ut.log_bernoulli_with_logits(x, z
        # 计算 KL 散度，衡量编码分布与先验分布之间的差异
        kl = torch.mean(ut.kl_normal(m, v, self.z_prior_m,
        # 计算负 ELBO（Negative Evidence Lower Bound），即 KL
        nelbo = kl + rec


#################################################################
        # 代码修改结束

#################################################################
```

```
        return nelbo, kl, rec
```

## 3. To test your implementation, run python run_vae.py to train the VAE. Once the run is complete (20000 iterations), it will output (assuming your implementation is correct): the average (1) negative ELBO, (2) KL term, and (3) reconstruction loss as evaluated on a test subset that we have selected. Report the three numbers you obtain as part of the write-up. Since we' re using stochastic optimization, you may wish to run the model multiple times and report each metric' s mean and corresponding standard error. (Hint: the negative ELBO on the test subset should be somewhere around 100.)

## 运行代码结果


## 结果分析

训练的VAE在测试子集上的表现如下：

1. **Negative ELBO (NELBO):** 平均为 99.54
2. **KL Term:** 平均为 19.32
3. **Reconstruction Loss:** 平均为 80.21

这些结果表明，模型在负ELBO上表现良好，而KL散度和重构损失分别占据了负ELBO的一部分。在这个情况下，重构损失（Reconstruction Loss）的贡献较大，KL散度相对较小。最终的负ELBO约为100，这与提示中的期望值相符。

## 4. Visualize 200 digits (generate a single image tiled in a grid of $10 \times 20$ digits) sampled from $p_\theta(x)$.

随机采样的结果

迭代10000次结果

迭代20000次结果

可以看到迭代20000次结果较迭代10000次结果较为清晰,说明迭代20000训练出来的模型的效果越好，但是还存在个别数字比较难以辨认的问题。