基于图嵌入的高斯混合变分自编码器的深度聚类

Deep Clustering by Gaussian Mixture Variational Autoencoders with Graph Embedding, DGG

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

1. 引言

这篇博文主要是对论文“Deep Clustering by Gaussian Mixture Variational Autoencoders with Graph Embedding”的整理总结，这篇文章将图嵌入与概率深度高斯混合模型相结合，使网络学习到符合全局模型和局部结构约束的强大特征表示。将样本作为图上的节点，并最小化它们的后验分布之间的加权距离，在这里使用Jenson-Shannon散度作为距离度量。

阅读这篇博文的前提条件是：了解高斯混合模型用于聚类的算法，了解变分推断与变分自编码器，进一步了解变分深度嵌入(VaDE)模型。在知道高斯混合模型(GMM)与变分自编码器(VAE)之后，VaDE实际上是将这两者结合起来的一个产物。与VAE相比，VaDE在公式推导中多了一个变量c。与GMM相比，变量c就相当于是GMM中的隐变量z，而隐层得到的特征z相当于原来GMM中的数据x。而基于图嵌入的高斯混合变分自编码器的深度聚类(DGG)模型可以看做在VAE的基础上结合了高斯混合模型与图嵌入来完成聚类过程，公式推导中同样增加了表示类别的变量c，同时，目标函数后面加了一项图嵌入的约束项。比起VaDE来说，可以理解为多了一个约束项——图嵌入，当然目标函数还是有所不同。

下面主要介绍DGG模型目标函数的数学推导过程。推导过程用到了概率论与数理统计的相关知识，更用到了VaDE模型推导里面的知识，如果想要深入了解推导过程，请先看变分深度嵌入(VaDE)模型的“前提公式”。

2. 目标函数的由来与转化

- 根据样本所提供的信息，对总体分布中的未知参数θ进行估值，其中一种方式是极大似然估计。设总体的概率函数为$p(x;\theta)$, $x_1, x_2, \ldots, x_N$是来自该总体的样本，将样本的联合概率函数看成θ的函数$L(\theta)$.

$$L(\hat{\theta}) = \max_{\theta\in\Theta} L(\theta)$$

$$L(\theta) = L(\theta; x_1, \cdots x_N) = p(x_1;\theta)\, p(x_2;\theta) \cdots p(x_N;\theta)$$

- 针对自编码模型，目标函数可以定义最大化为对数似然函数：

$$\max_{\phi,\theta} \sum_{i=1}^{N} \ln p_\theta(x_i)$$

- 现在DGG模型总体目标函数在上述基础上增加图嵌入的约束项：

$$\boxed{\max_{\phi,\theta} \sum_{i=1}^{N} \left(\ln p_\theta(x_i) - \sum_{j=1}^{N} w_{ij} JS(q_\phi(z,c \mid x_i), q_\phi(z,c \mid x_j)) \right)}$$

6. JS divergence between two Gaussian

We plot the JS divergence between two Gaussian distribution with varying relative orientation. From the figure, we can find that the JS divergence is minimized when the two Gaussian aligned with the coordinate.

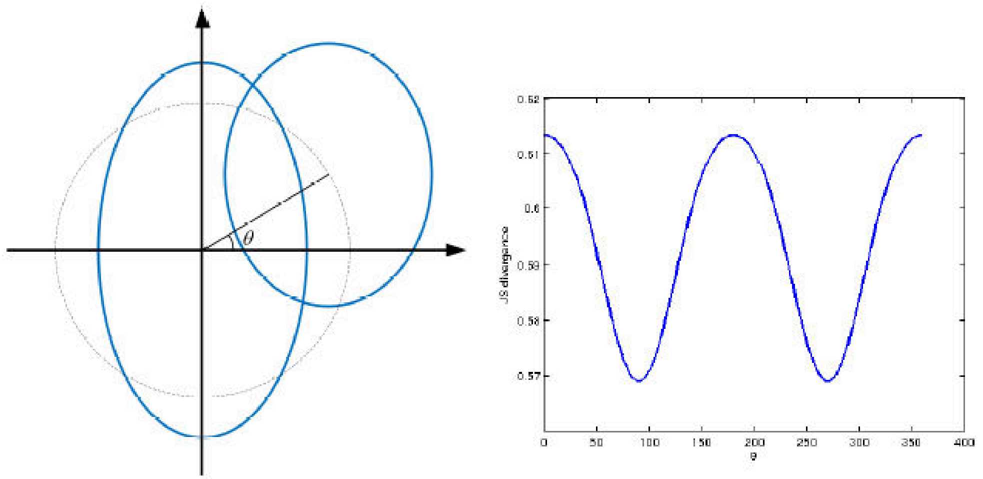


Figure 1. The JS divergence between two Gaussian distributions with different orientation. The two Gaussian distributions have diagonal covariance matrix, and the distance between their mean are fixed.

- DGG模型总体目标函数：

$$\max_{\phi,\theta}\sum_{i=1}^{N}\left(\ln p_\theta(x_i)-\sum_{j=1}^{N}w_{ij}JS(q_\phi(z,c\mid x_i),q_\phi(z,c\mid x_j))\right)$$

$$\overset{(1)}{\geq}\sum_{i=1}^{N}\left(\ln p_\theta(x_i)-\sum_{j=1}^{N}w_{ij}G(\phi,\theta,x_i,x_j)\right)$$

$$\overset{(2)}{=}\sum_{i=1}^{N}\left(\ln p_\theta(x_i)-\sum_{j=1}^{N}w_{ij}\ln p_\theta(x_i)\right)+\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}(L(\theta,\phi;x_i)+L(\theta,\phi;x_i,x_j))$$

$$\overset{(3)}{=}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}(L(\theta,\phi;x_i)+L(\theta,\phi;x_i,x_j))$$

- DGG目标函数最终转化为：

$$\boxed{\max_{\phi,\theta}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}(L(\theta,\phi;x_i)+L(\theta,\phi;x_i,x_j))}$$

注：步骤(1)(2)(3)的推导见下三页

DGG转化的目标函数的详细推导——(1)

(1) 定义一个辅助函数G()

$$令M=\frac{1}{2}(q_\phi(z,c|x_i)+q_\phi(z,c|x_j))$$

$$\begin{aligned}
G(\phi,\theta,x_i,x_j) &= \frac{1}{2}KL(q_\phi(z,c|x_i)\,||\,p_\theta(z,c|x_i))+\frac{1}{2}KL(q_\phi(z,c|x_j)\,||\,p_\theta(z,c|x_i)) \\
&= \frac{1}{2}\int_z\sum_c q_\phi(z,c|x_i)\ln\frac{q_\phi(z,c|x_i)}{p_\theta(z,c|x_i)}dz+\frac{1}{2}\int_z\sum_c q_\phi(z,c|x_j)\ln\frac{q_\phi(z,c|x_j)}{p_\theta(z,c|x_i)}dz \\
&= \frac{1}{2}\int_z\sum_c q_\phi(z,c|x_i)\ln\frac{q_\phi(z,c|x_i)}{M}dz+\frac{1}{2}\int_z\sum_c q_\phi(z,c|x_j)\ln\frac{q_\phi(z,c|x_j)}{M}dz \\
&\quad +\frac{1}{2}\int_z\sum_c (q_\phi(z,c|x_i)+q_\phi(z,c|x_j))\ln\frac{M}{p_\theta(z,c|x_i)}dz \\
&= JS(q_\phi(z,c|x_i)\,||\,q_\phi(z,c|x_j))+KL(M\,||\,p_\theta(z,c|x_i)) \\
&\geq JS(q_\phi(z,c|x_i)\,||\,q_\phi(z,c|x_j))
\end{aligned}$$

注：KL散度恒大于等于0.

DGG转化的目标函数的详细推导——(2)

(2) 原先推导VAE的目标函数时，用到的式子：

$$\ln p_\theta(x) = KL(q_\phi(z|x)\,\|\,p_\theta(z|x)) + \boxed{E_{q_\phi(z|x_i)}\left[\ln \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]}$$

VAE的目标函数

现在DGG模型，首先引入一个类别变量c：

$$\ln p_\theta(x_i) = KL(q_\phi(z,c|x_i)\,\|\,p_\theta(z,c|x_i)) + E_{q_\phi(z,c|x_i)}\left[\ln \frac{p_\theta(x_i,z,c)}{q_\phi(z,c|x_i)}\right]$$

$$\overset{(2.1)}{=} KL(q_\phi(z,c|x_j)\,\|\,p_\theta(z,c|x_i)) + E_{q_\phi(z,c|x_j)}\left[\ln \frac{p_\theta(x_i,z,c)}{q_\phi(z,c|x_j)}\right]$$

$$\overset{(2.2)}{=} \frac{1}{2}\Big(KL(q_\phi(z,c|x_i)\,\|\,p_\theta(z,c|x_i)) + KL(q_\phi(z,c|x_j)\,\|\,p_\theta(z,c|x_i))\Big)$$

$$+\frac{1}{2}\left(E_{q_\phi(z,c|x_i)}\left[\ln \frac{p_\theta(x_i,z,c)}{q_\phi(z,c|x_i)}\right] + E_{q_\phi(z,c|x_j)}\left[\ln \frac{p_\theta(x_i,z,c)}{q_\phi(z,c|x_j)}\right]\right)$$

$$\overset{(2.3)}{=} G(\phi,\theta,x_i,x_j) + \frac{1}{2}\big(L(\theta,\phi;x_i) + L(\theta,\phi;x_i,x_j)\big)$$

DGG转化的目标函数的详细推导——(2.1)(2.2)

(2.1) 推导过程

$$KL(q_\phi(z,c|x_j)\,\|\,p_\theta(z,c|x_i)) + E_{q_\phi(z,c|x_j)}\left[\ln\frac{p_\theta(x_i,z,c)}{q_\phi(z,c|x_j)}\right]$$

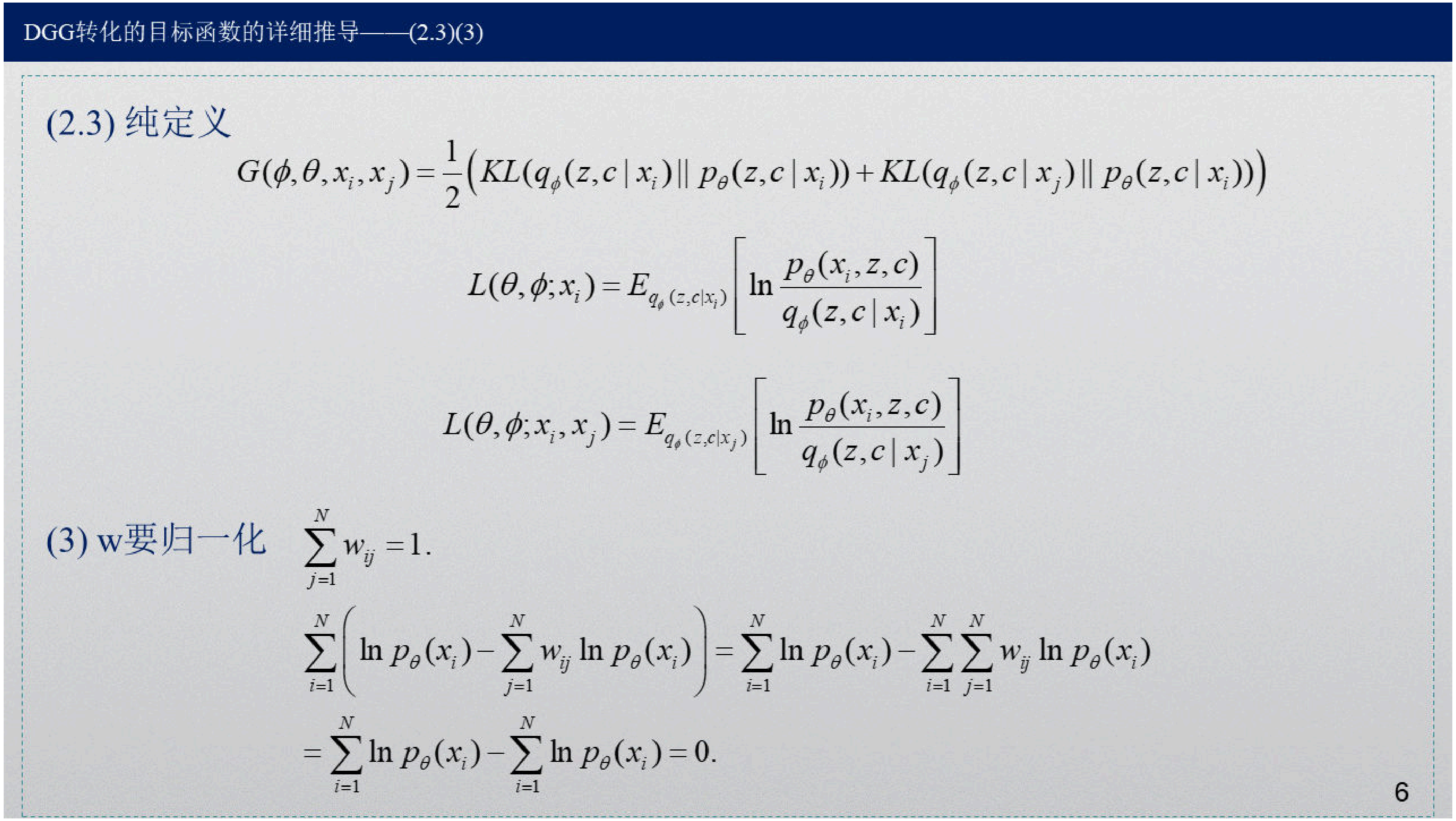
$$= -\int_z\sum_c q_\phi(z,c|x_j)\ln\frac{p_\theta(z,c|x_i)}{q_\phi(z,c|x_j)}dz + \int_z\sum_c q_\phi(z,c|x_j)\ln\frac{p_\theta(x_i,z,c)}{q_\phi(z,c|x_j)}dz$$

$$= \int_z\sum_c q_\phi(z,c|x_j)\left(\ln\frac{p_\theta(x_i,z,c)}{q_\phi(z,c|x_j)} - \ln\frac{p_\theta(z,c|x_i)}{q_\phi(z,c|x_j)}\right)dz$$

$$= \int_z\sum_c q_\phi(z,c|x_j)\left(\ln\frac{p_\theta(x_i,z,c)q_\phi(z,c|x_j)}{q_\phi(z,c|x_j)p_\theta(z,c|x_i)}\right)dz$$

$$= \int_z\sum_c q_\phi(z,c|x_j)\ln p_\theta(x_i)dz = \ln p_\theta(x_i)\int_z\sum_c q_\phi(z,c|x_j)dz = \ln p_\theta(x_i)$$

(2.2) 将上两行式子加起来除以2即可得到.



DGG转化的目标函数的详细推导——(2.3)(3)

(2.3) 纯定义

$$G(\phi,\theta,x_i,x_j)=\frac{1}{2}\Big(KL(q_\phi(z,c\,|\,x_i)\,\|\,p_\theta(z,c\,|\,x_i))+KL(q_\phi(z,c\,|\,x_j)\,\|\,p_\theta(z,c\,|\,x_i))\Big)$$

$$L(\theta,\phi;x_i)=E_{q_\phi(z,c|x_i)}\left[\ln\frac{p_\theta(x_i,z,c)}{q_\phi(z,c\,|\,x_i)}\right]$$

$$L(\theta,\phi;x_i,x_j)=E_{q_\phi(z,c|x_j)}\left[\ln\frac{p_\theta(x_i,z,c)}{q_\phi(z,c\,|\,x_j)}\right]$$

(3) w要归一化 $\sum_{j=1}^{N}w_{ij}=1.$

$$\sum_{i=1}^{N}\left(\ln p_\theta(x_i)-\sum_{j=1}^{N}w_{ij}\ln p_\theta(x_i)\right)=\sum_{i=1}^{N}\ln p_\theta(x_i)-\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}\ln p_\theta(x_i)$$

$$=\sum_{i=1}^{N}\ln p_\theta(x_i)-\sum_{i=1}^{N}\ln p_\theta(x_i)=0.$$

6

3．目标函数具体推导

- DGG模型总体目标函数：

$$\boxed{\max_{\phi,\theta} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(L(\theta,\phi;x_i)+L(\theta,\phi;x_i,x_j))}$$

- 推断模型：

假设$\boxed{q_\phi(z,c|x)=q_{\phi_1}(z|x)q_{\phi_2}(c|z)}$

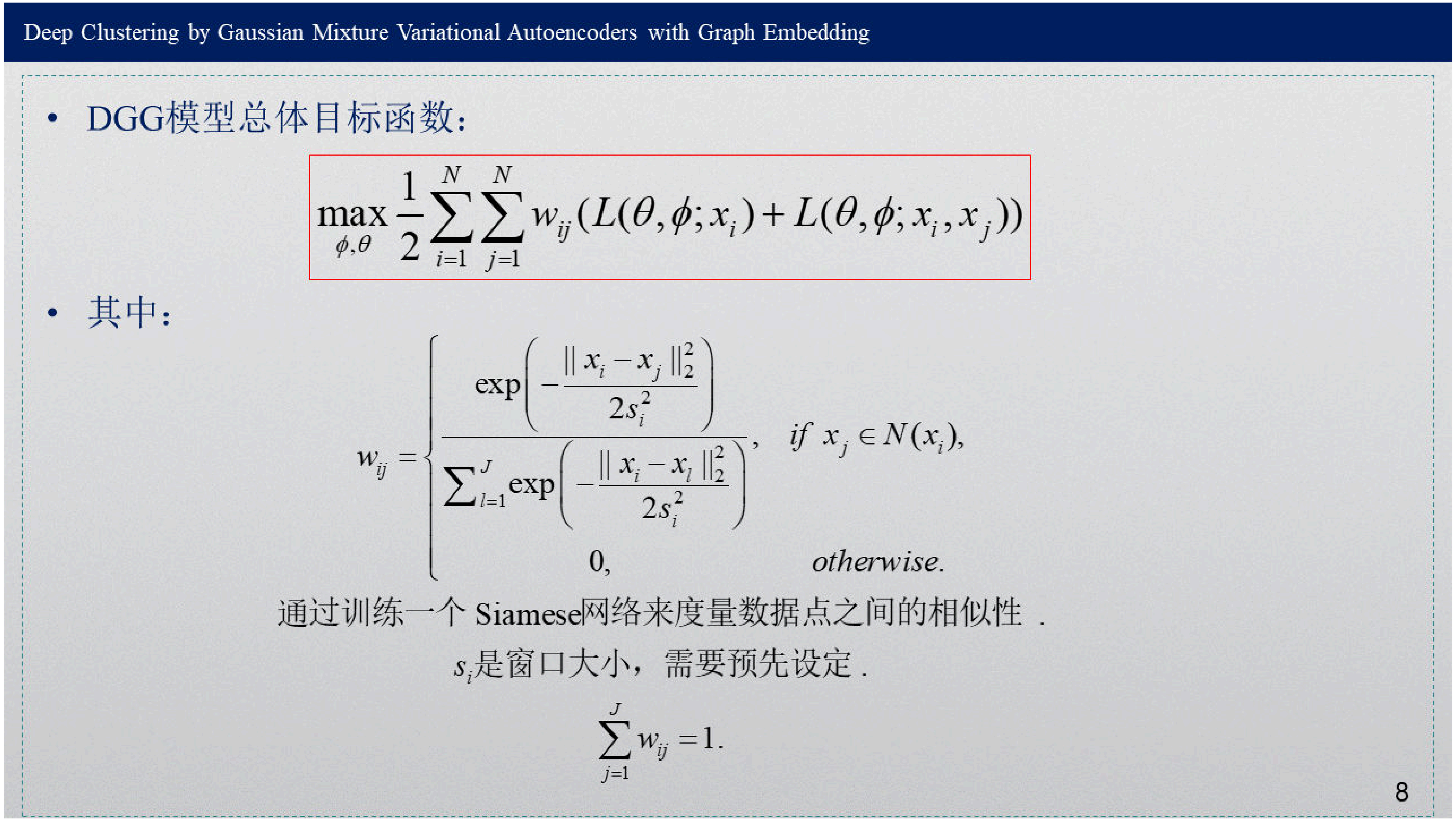
定义：$q_{\phi_1}(z|x)=N(\tilde{\mu},diag(\tilde{\sigma}^2))$, $q_{\phi_2}(c|z)=Multinomial(\tilde{\pi})$, $\gamma_{ik}=q(c_k=1|z_i)$

其中：$[\tilde{\mu},\log(\tilde{\sigma}^2)]=f_1(x;\phi_1)$(编码器)，$\tilde{\pi}=f_2(z;\phi_2)$(分类器)，$z=\tilde{\mu}+\tilde{\sigma}\circ\varepsilon$

- 生成模型：$p(c)=p(c_1,\cdots c_K)=\prod_{k=1}^{K}\pi_k^{c_k}$, $p(c_k=1)=\pi_{ik}$, $p_\theta(z|c_k=1)=N(\mu_k,diag(\sigma_k^2))$

$$p_\theta(x|z)=\begin{cases} Ber(\mu_x), & if\ x\ is\ binary, \\ N(\mu_x,\lambda I), & if\ x\ is\ real\text{-}valued. \end{cases}$$

$\mu_x=g(z;\theta)$(解码器)



- DGG模型总体目标函数：

$$\boxed{\max_{\phi,\theta}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}(L(\theta,\phi;x_i)+L(\theta,\phi;x_i,x_j))}$$

- 其中：

$$w_{ij}=\begin{cases}\dfrac{\exp\left(-\dfrac{\|x_i-x_j\|_2^2}{2s_i^2}\right)}{\sum_{l=1}^{J}\exp\left(-\dfrac{\|x_i-x_l\|_2^2}{2s_i^2}\right)}, & if\ x_j\in N(x_i),\\ 0, & otherwise.\end{cases}$$

通过训练一个 Siamese网络来度量数据点之间的相似性 .

s_i是窗口大小，需要预先设定 .

$$\sum_{j=1}^{J}w_{ij}=1.$$

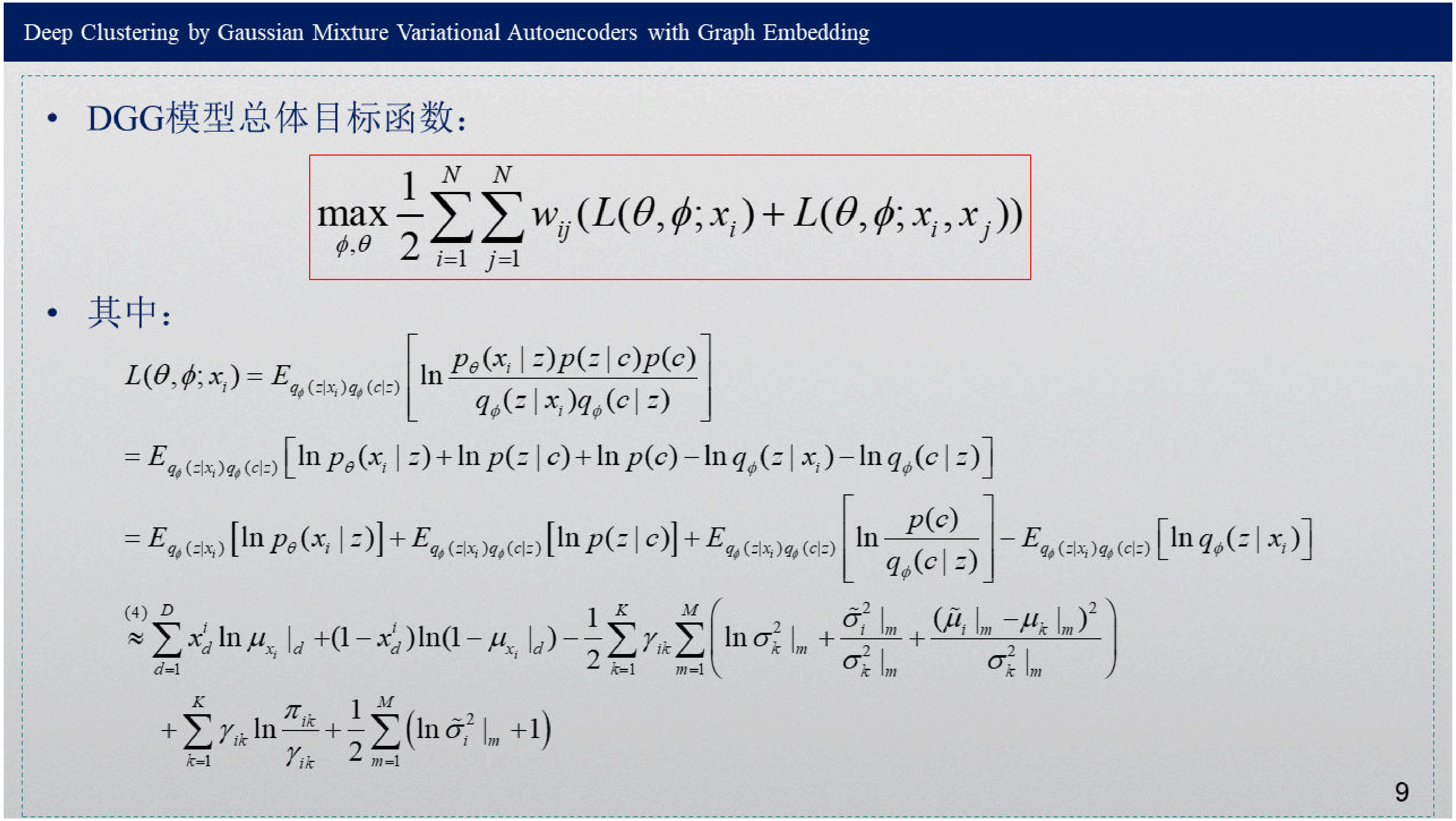
用Siamese网络来度量数据点之间的相似度，从而选出数据点xi的邻居。

5. Details of the Siamese Network

We train a Siamese network to measure the similarity between the data points. We select N_t nearest neighbors for each datapoint, and group them into N_t pairs. We label these pairs as as positive. We then randomly select N_t pairs of data and treat them as negative. We then minimize the following contrastive loss

$$L = \begin{cases} \|x_i - x_j\|_2^2 & (x_i, x_j) \text{ is positive pair} \\ \max(c - \|x_i - x_j\|_2, 0)^2 & (x_i, x_j) \text{ is negative pair} \end{cases} \tag{19}$$

where c is a predefined parameter. In our experiments, the architecture of the Siamese network is set to same with that of encoder, i.e., $D - 500 - 500 - 2000 - 10$. The network is fully connected, and ReLU is used as the activation function. The parameter c is set to 3. N_t is set to 2, 5, 3, 3 for MNIST, STL-10, Reuters, and HHAR, respectively. The network is trained using Adam optimizer with the initial learning rate 0.0005. The learning rate decays every 70 epochs with a factor 0.1. The weight decay in Adam is set to 0.0001



Deep Clustering by Gaussian Mixture Variational Autoencoders with Graph Embedding

- DGG模型总体目标函数：

$$\boxed{\max_{\phi,\theta} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\left(L(\theta,\phi;x_i)+L(\theta,\phi;x_i,x_j)\right)}$$

- 其中：

$$\begin{aligned}
L(\theta,\phi;x_i) &= E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln\frac{p_\theta(x_i\,|\,z)p(z\,|\,c)p(c)}{q_\phi(z\,|\,x_i)q_\phi(c\,|\,z)}\right]\\
&= E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln p_\theta(x_i\,|\,z)+\ln p(z\,|\,c)+\ln p(c)-\ln q_\phi(z\,|\,x_i)-\ln q_\phi(c\,|\,z)\right]\\
&= E_{q_\phi(z|x_i)}\left[\ln p_\theta(x_i\,|\,z)\right]+E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln p(z\,|\,c)\right]+E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln\frac{p(c)}{q_\phi(c\,|\,z)}\right]-E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln q_\phi(z\,|\,x_i)\right]\\
&\overset{(4)}{\approx} \sum_{d=1}^{D} x_d^i \ln \mu_{x_i}|_d + (1-x_d^i)\ln(1-\mu_{x_i}|_d) - \frac{1}{2}\sum_{k=1}^{K}\gamma_{ik}\sum_{m=1}^{M}\left(\ln\sigma_k^2|_m + \frac{\tilde{\sigma}_i^2|_m}{\sigma_k^2|_m} + \frac{(\tilde{\mu}_i|_m - \mu_k|_m)^2}{\sigma_k^2|_m}\right)\\
&\quad + \sum_{k=1}^{K}\gamma_{ik}\ln\frac{\pi_{ik}}{\gamma_{ik}} + \frac{1}{2}\sum_{m=1}^{M}\left(\ln\tilde{\sigma}_i^2|_m + 1\right)
\end{aligned}$$

其中，D是数据x的维度，M是隐层z的维度，K是聚类数。

- DGG模型总体目标函数：

$$\boxed{\max_{\phi,\theta} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(L(\theta,\phi;x_i)+L(\theta,\phi;x_i,x_j))}$$

- 其中：

$$L(\theta,\phi;x_i,x_j)=E_{q_\phi(z|x_j)q_\phi(c|z)}\left[\ln\frac{p_\theta(x_i\mid z)p(z\mid c)p(c)}{q_\phi(z\mid x_j)q_\phi(c\mid z)}\right]$$

$$=E_{q_\phi(z|x_j)q_\phi(c|z)}\left[\ln p_\theta(x_i\mid z)+\ln p(z\mid c)+\ln p(c)-\ln q_\phi(z\mid x_j)-\ln q_\phi(c\mid z)\right]$$

$$=E_{q_\phi(z|x_j)}\left[\ln p_\theta(x_i\mid z)\right]+E_{q_\phi(z|x_j)q_\phi(c|z)}\left[\ln p(z\mid c)\right]+E_{q_\phi(z|x_j)q_\phi(c|z)}\left[\ln\frac{p(c)}{q_\phi(c\mid z)}\right]-E_{q_\phi(z|x_j)q_\phi(c|z)}\left[\ln q_\phi(z\mid x_j)\right]$$

$$\overset{(5)}{\approx}\sum_{d=1}^{D}x_d^i\ln\mu_{x_j}|_d+(1-x_d^i)\ln(1-\mu_{x_j}|_d)-\frac{1}{2}\sum_{k=1}^{K}\gamma_{jk}\sum_{m=1}^{M}\left(\ln\sigma_k^2|_m+\frac{\tilde{\sigma}_j^2|_m}{\sigma_k^2|_m}+\frac{(\tilde{\mu}_j|_m-\mu_k|_m)^2}{\sigma_k^2|_m}\right)$$

$$+\sum_{k=1}^{K}\gamma_{jk}\ln\frac{\pi_{ik}}{\gamma_{jk}}+\frac{1}{2}\sum_{m=1}^{M}\left(\ln\tilde{\sigma}_j^2|_m+1\right)$$

DGG目标函数继续推导——(4)

$$L(\theta,\phi;x_i)=\boxed{E_{q_\phi(z|x_i)}[\ln p_\theta(x_i|z)]+E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln p(z|c)]}+E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln\frac{p(c)}{q_\phi(c|z)}\right]-E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln q_\phi(z|x_i)]$$

(4)第一项

$$E_{q_\phi(z|x_i)}[\ln p_\theta(x_i|z)]\approx\frac{1}{L}\sum_{l=1}^{L}\ln p_\theta(x|z^{(l)}).\quad 令L=1,$$

$$E_{q_\phi(z|x_i)}[\ln p_\theta(x_i|z)]\approx\begin{cases}\sum_{d=1}^{D}x_d^i\ln\mu_{x_i}|_d+(1-x_d^i)\ln(1-\mu_{x_i}|_d), & if\ x_i\ is\ binary,\\ \frac{1}{2\lambda}\|x_i-\mu_{x_i}\|_2^2, & if\ x_i\ is\ real\text{-}valued.\end{cases}$$

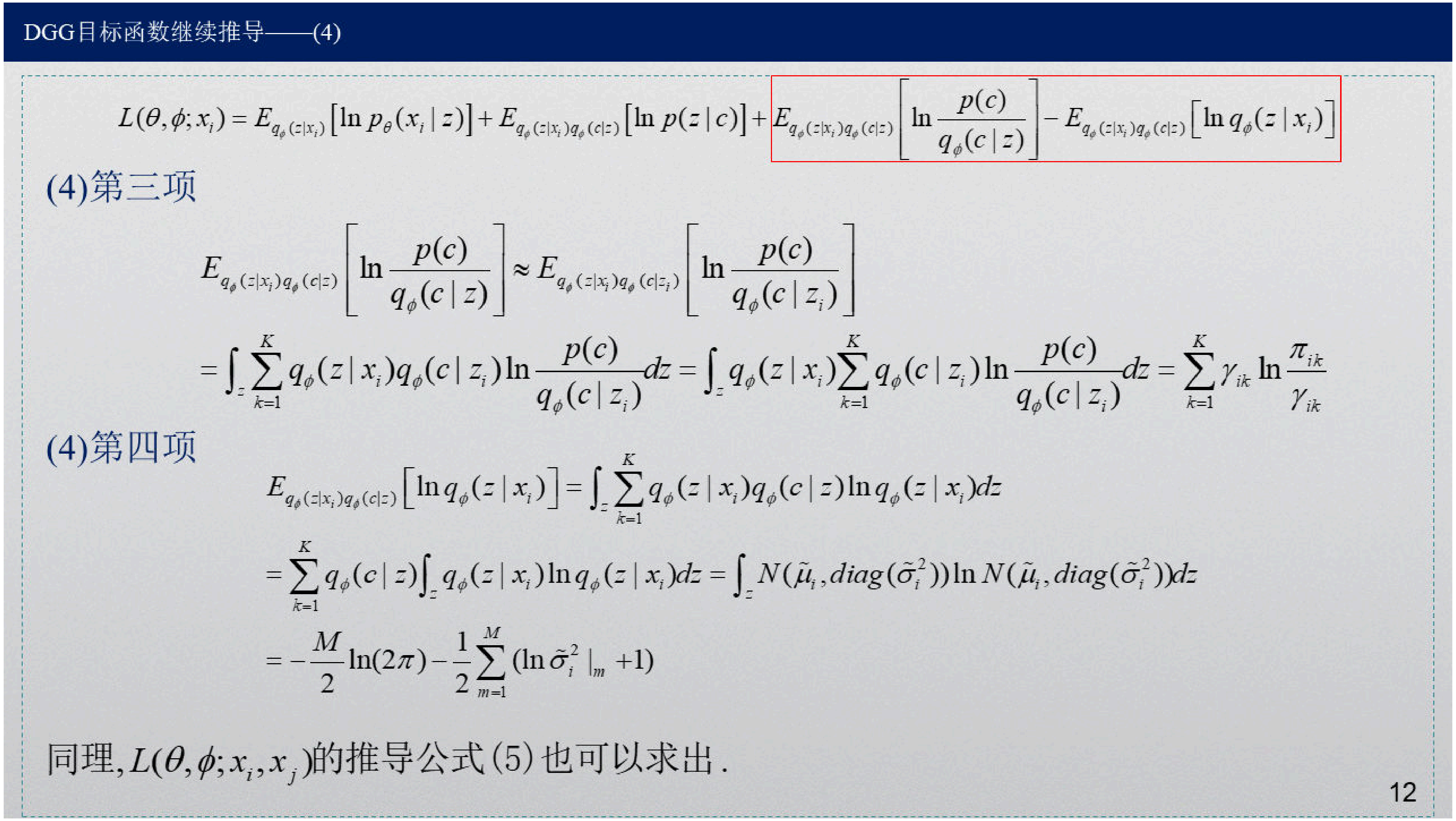
(4)第二项

$$E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln p(z|c)]\approx E_{q_\phi(z|x_i)q_\phi(c|z_i)}[\ln p(z|c)]$$

$$=\int_z\sum_{k=1}^{K}q_\phi(z|x_i)q_\phi(c|z_i)\ln p(z|c)dz=\sum_{k=1}^{K}q_\phi(c|z_i)\int_z q_\phi(z|x_i)\ln p(z|c)dz$$

$$=\sum_{k=1}^{K}\gamma_{ik}\int_z N(\tilde{\mu}_i,diag(\tilde{\sigma}_i^2))\ln N(\mu_k,diag(\sigma_k^2))dz$$

$$=-\frac{1}{2}\sum_{k=1}^{K}\gamma_{ik}\left[M\ln(2\pi)+\sum_{m=1}^{M}\left(\ln\sigma_k^2|_m+\frac{\tilde{\sigma}_i^2|_m}{\sigma_k^2|_m}+\frac{(\tilde{\mu}_i|_m-\mu_k|_m)^2}{\sigma_k^2|_m}\right)\right]$$


DGG目标函数继续推导——(4)

$$L(\theta,\phi;x_i)=E_{q_\phi(z|x_i)}[\ln p_\theta(x_i|z)]+E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln p(z|c)]+\boxed{E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln\frac{p(c)}{q_\phi(c|z)}\right]-E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln q_\phi(z|x_i)]}$$

(4)第三项

$$E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln\frac{p(c)}{q_\phi(c|z)}\right]\approx E_{q_\phi(z|x_i)q_\phi(c|z_i)}\left[\ln\frac{p(c)}{q_\phi(c|z_i)}\right]$$

$$=\int_z\sum_{k=1}^K q_\phi(z|x_i)q_\phi(c|z_i)\ln\frac{p(c)}{q_\phi(c|z_i)}dz=\int_z q_\phi(z|x_i)\sum_{k=1}^K q_\phi(c|z_i)\ln\frac{p(c)}{q_\phi(c|z_i)}dz=\sum_{k=1}^K\gamma_{ik}\ln\frac{\pi_{ik}}{\gamma_{ik}}$$

(4)第四项

$$E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln q_\phi(z|x_i)]=\int_z\sum_{k=1}^K q_\phi(z|x_i)q_\phi(c|z)\ln q_\phi(z|x_i)dz$$

$$=\sum_{k=1}^K q_\phi(c|z)\int_z q_\phi(z|x_i)\ln q_\phi(z|x_i)dz=\int_z N(\tilde{\mu}_i,diag(\tilde{\sigma}_i^2))\ln N(\tilde{\mu}_i,diag(\tilde{\sigma}_i^2))dz$$

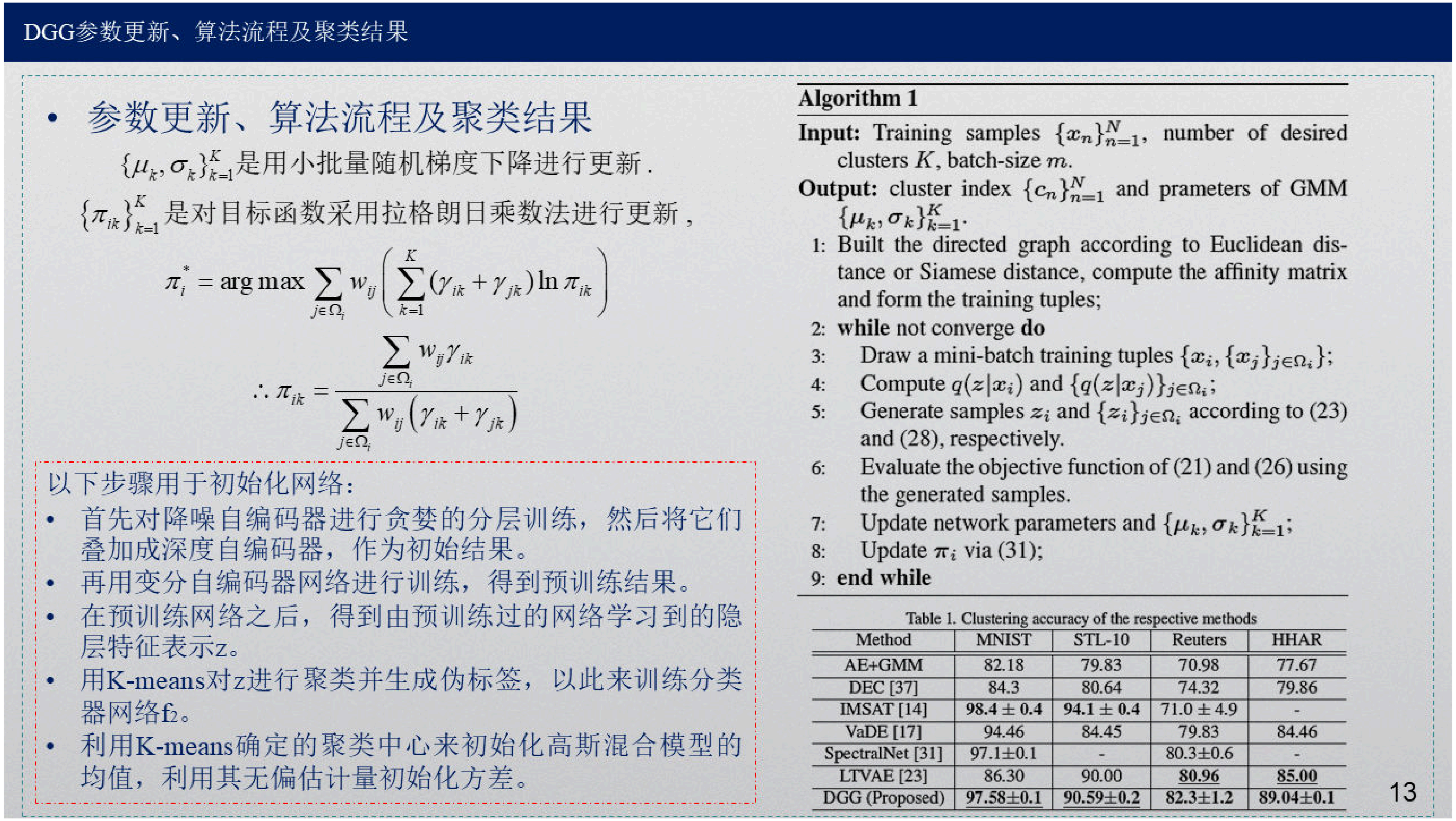
$$=-\frac{M}{2}\ln(2\pi)-\frac{1}{2}\sum_{m=1}^M(\ln\tilde{\sigma}_i^2|_m+1)$$

同理, $L(\theta,\phi;x_i,x_j)$的推导公式(5)也可以求出.

12

第二项和第四项最后一步怎么来的？用到了一个公式，公式的具体推导见：变分深度嵌入(VaDE)模型的"前提公式"。

4. 参数更新过程及聚类结果



DGG参数更新、算法流程及聚类结果

- 参数更新、算法流程及聚类结果

$\{\mu_k, \sigma_k\}_{k=1}^{K}$是用小批量随机梯度下降进行更新.

$\{\pi_{ik}\}_{k=1}^{K}$是对目标函数采用拉格朗日乘数法进行更新,

$$\pi_i^* = \arg\max \sum_{j\in\Omega_i} w_{ij}\left(\sum_{k=1}^{K}(\gamma_{ik}+\gamma_{jk})\ln\pi_{ik}\right)$$

$$\therefore \pi_{ik} = \frac{\sum_{j\in\Omega_i} w_{ij}\gamma_{ik}}{\sum_{j\in\Omega_i} w_{ij}\left(\gamma_{ik}+\gamma_{jk}\right)}$$

以下步骤用于初始化网络:

- 首先对降噪自编码器进行贪婪的分层训练，然后将它们叠加成深度自编码器，作为初始结果。
- 再用变分自编码器网络进行训练，得到预训练结果。
- 在预训练网络之后，得到由预训练过的网络学习到的隐层特征表示z。
- 用K-means对z进行聚类并生成伪标签，以此来训练分类器网络f₂。
- 利用K-means确定的聚类中心来初始化高斯混合模型的均值，利用其无偏估计量初始化方差。

Algorithm 1

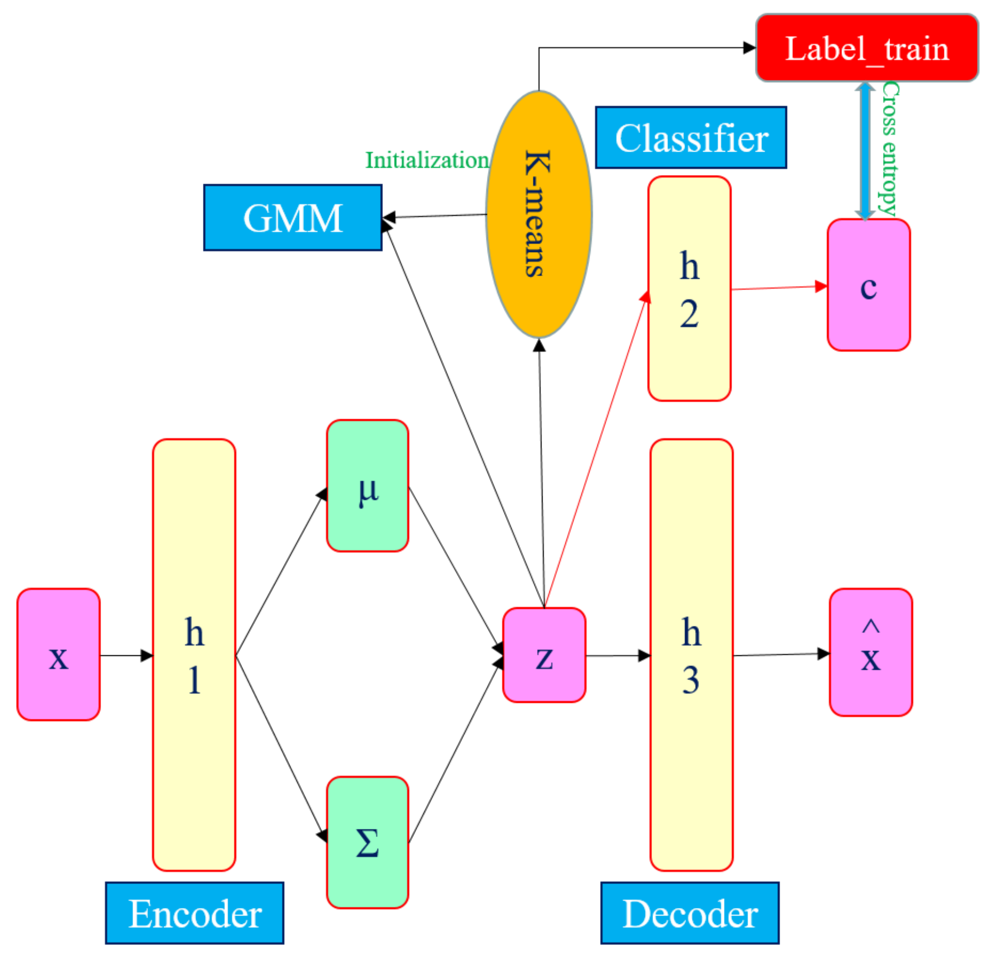
```
Input: Training samples $\{x_n\}_{n=1}^{N}$, number of desired
    clusters $K$, batch-size $m$.
Output: cluster index $\{c_n\}_{n=1}^{N}$ and prameters of GMM
    $\{\mu_k, \sigma_k\}_{k=1}^{K}$.
1: Built the directed graph according to Euclidean dis-
   tance or Siamese distance, compute the affinity matrix
   and form the training tuples;
2: while not converge do
3:     Draw a mini-batch training tuples $\{x_i, \{x_j\}_{j\in\Omega_i}\}$;
4:     Compute $q(z|x_i)$ and $\{q(z|x_j)\}_{j\in\Omega_i}$;
5:     Generate samples $z_i$ and $\{z_i\}_{j\in\Omega_i}$ according to (23)
       and (28), respectively.
6:     Evaluate the objective function of (21) and (26) using
       the generated samples.
7:     Update network parameters and $\{\mu_k, \sigma_k\}_{k=1}^{K}$;
8:     Update $\pi_i$ via (31);
9: end while
```

Table 1. Clustering accuracy of the respective methods

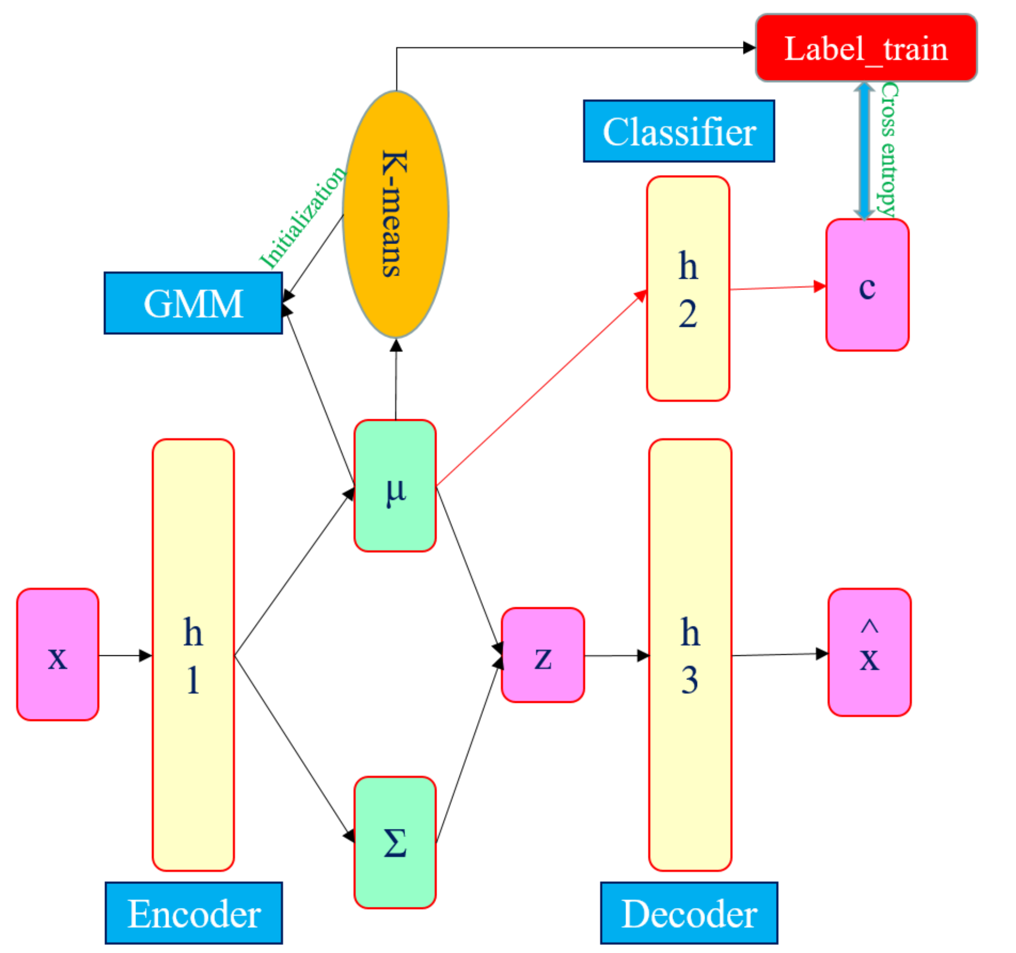
Method	MNIST	STL-10	Reuters	HHAR
AE+GMM	82.18	79.83	70.98	77.67
DEC [37]	84.3	80.64	74.32	79.86
IMSAT [14]	**98.4 ± 0.4**	**94.1 ± 0.4**	71.0 ± 4.9	-
VaDE [17]	94.46	84.45	79.83	84.46
SpectralNet [31]	97.1±0.1	-	80.3±0.6	-
LTVAE [23]	86.30	90.00	80.96	85.00
DGG (Proposed)	**97.58±0.1**	**90.59±0.2**	**82.3±1.2**	**89.04±0.1**

13

我也搞不明白哪个图正确，或者都不正确，望指正。



看完论文后以为的图



看完代码后理解的图

5. 我的思考

- 在推导过程中我与原文中的推导有不一样的地方。

1）我的推导过程中变分下界L中第二项系数是1/2，原文直接是1，而在支撑材料里面仍然是1/2，因此可以认为是作者笔误造成的。

2）我的推导过程中变分下界L中的第二项与第四项都有常数项（蓝框框标出的），这两项正好正负抵消，才没有这个常数项，而在原文支撑材料里面直接第二四项都没有常数项。不过这只是支撑材料的内容，在原文中没有太大影响。

推导不一致的地方

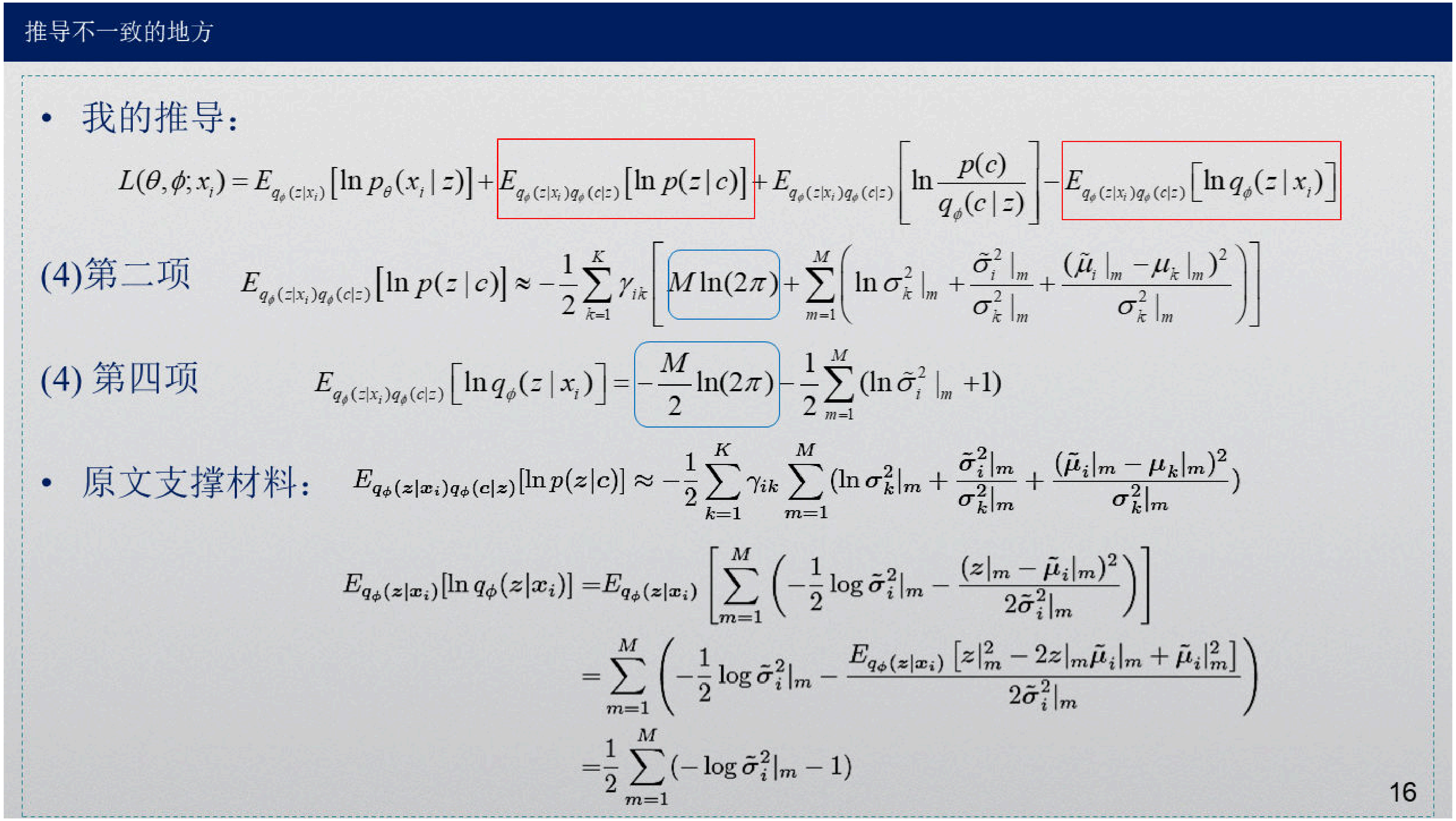
- 我的推导：

$$L(\theta,\phi;x_i)=E_{q_\phi(z|x_i)}\left[\ln p_\theta(x_i\mid z)\right]+E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln p(z\mid c)\right]+E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln\frac{p(c)}{q_\phi(c\mid z)}\right]-E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln q_\phi(z\mid x_i)\right]$$

$$\overset{(4)}{\approx}\sum_{d=1}^{D}x_d^i\ln\mu_{x_i}|_d+(1-x_d^i)\ln(1-\mu_{x_i}|_d)\boxed{-\frac{1}{2}}\sum_{k=1}^{K}\gamma_{ik}\sum_{m=1}^{M}\left(\ln\sigma_k^2|_m+\frac{\tilde{\sigma}_i^2|_m}{\sigma_k^2|_m}+\frac{(\tilde{\mu}_i|_m-\mu_k|_m)^2}{\sigma_k^2|_m}\right)+\sum_{k=1}^{K}\gamma_{ik}\ln\frac{\pi_{ik}}{\gamma_{ik}}+\frac{1}{2}\sum_{m=1}^{M}\left(\ln\tilde{\sigma}_i^2|_m+1\right)$$

- 原文中的叙述：

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x}_i)\\
&\approx\sum_{d=1}^{D}x_d^i\log\boldsymbol{\mu}_{x_i}|_d+(1-x_d^i)\log(1-\boldsymbol{\mu}_{x_i}|_d)\\
&\quad\boxed{-}\sum_{k=1}^{K}\gamma_{ik}\sum_{m=1}^{M}(\log\boldsymbol{\sigma}_k^2|_m+\frac{\tilde{\boldsymbol{\sigma}}_i^2|_m}{\boldsymbol{\sigma}_k^2|_m}+\frac{(\tilde{\boldsymbol{\mu}}_i|_m-\boldsymbol{\mu}_k|_m)^2}{\boldsymbol{\sigma}_k^2|_m})\\
&\quad+\sum_{k=1}^{K}\gamma_{ik}\log\frac{\pi_{ik}}{\gamma_{ik}}+\frac{1}{2}\sum_{m=1}^{M}(1+\log\tilde{\boldsymbol{\sigma}}_i^2|_m)
\end{aligned}\tag{21}$$



推导不一致的地方

- 我的推导：

$$L(\theta,\phi;x_i)=E_{q_\phi(z|x_i)}[\ln p_\theta(x_i|z)]+\boxed{E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln p(z|c)]}+E_{q_\phi(z|x_i)q_\phi(c|z)}\left[\ln\frac{p(c)}{q_\phi(c|z)}\right]-\boxed{E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln q_\phi(z|x_i)]}$$

(4)第二项 $$E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln p(z|c)]\approx-\frac{1}{2}\sum_{k=1}^{K}\gamma_{ik}\left[\boxed{M\ln(2\pi)}+\sum_{m=1}^{M}\left(\ln\sigma_k^2|_m+\frac{\tilde{\sigma}_i^2|_m}{\sigma_k^2|_m}+\frac{(\tilde{\mu}_i|_m-\mu_k|_m)^2}{\sigma_k^2|_m}\right)\right]$$

(4) 第四项 $$E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln q_\phi(z|x_i)]=\boxed{-\frac{M}{2}\ln(2\pi)}-\frac{1}{2}\sum_{m=1}^{M}(\ln\tilde{\sigma}_i^2|_m+1)$$

- 原文支撑材料： $$E_{q_\phi(z|x_i)q_\phi(c|z)}[\ln p(z|c)]\approx-\frac{1}{2}\sum_{k=1}^{K}\gamma_{ik}\sum_{m=1}^{M}(\ln\sigma_k^2|_m+\frac{\tilde{\sigma}_i^2|_m}{\sigma_k^2|_m}+\frac{(\tilde{\mu}_i|_m-\mu_k|_m)^2}{\sigma_k^2|_m})$$

$$\begin{aligned}E_{q_\phi(z|x_i)}[\ln q_\phi(z|x_i)] &=E_{q_\phi(z|x_i)}\left[\sum_{m=1}^{M}\left(-\frac{1}{2}\log\tilde{\sigma}_i^2|_m-\frac{(z|_m-\tilde{\mu}_i|_m)^2}{2\tilde{\sigma}_i^2|_m}\right)\right]\\ &=\sum_{m=1}^{M}\left(-\frac{1}{2}\log\tilde{\sigma}_i^2|_m-\frac{E_{q_\phi(z|x_i)}\left[z|_m^2-2z|_m\tilde{\mu}_i|_m+\tilde{\mu}_i|_m^2\right]}{2\tilde{\sigma}_i^2|_m}\right)\\ &=\frac{1}{2}\sum_{m=1}^{M}(-\log\tilde{\sigma}_i^2|_m-1)\end{aligned}$$

16

- 这里有一点和VaDE不一样，VaDE聚类结果是由${{\gamma }_{ik}}$后验概率通过贝叶斯公式得到的，中间过程并没有参与梯度下降的更新，而DGG这里是构建了一个分类器网络f_2，从而得到聚类结果。

  1）DGG这里有三个网络，f_1是编码器，g是解码器，而f_2是分类器。

  2）这种架构和苏剑林博客中提到的VAE用于聚类的算法"变分自编码器（四）：一步到位的聚类方案 - 科学空间|Scientific Spaces"的网络架构有异曲同工之妙，不过苏剑林博客中的网路框架还多了一个自定义的Gaussian层，有兴趣的可以看看苏剑林那篇文章及代码。

- 原文中分类器的输入应该是VAE模型得到的隐层z，而代码里面是VAE得到的x_mean，没有经过采样得到z，直接用x_mean作为分类器的输入，这一点不知道是我理解有误，还是代码问题。

- 关于预训练：首先训练DAE作为VAE的初始化，训练VAE得到隐层参数${\mu}_{i}$，用K-means对${\mu}_{i}$进行聚类，得到聚类中心作为GMM的初始类均值，K-means得到类标签，计算样本方差作为GMM初始类协方差矩阵。分类器部分用SGD进行微调，将${{\gamma }_{ik}}$作为分类器输出，并用${{\gamma }_{ik}}$得到的聚类结果与K-means得到的初始标签之间的交叉熵作为分类器的损失函数，从而预训练分类器的参数。
- 关于Siamese网络找邻居：通过神经网络训练得到10维的隐层特征，对隐层计算相似性，代码中是用Siamese网络找到数据点的前100个邻居，但后来又从这100个邻居中随机取20个作为数据点的20个邻居，为什么不是先对100个邻居排序，取最近的前20个？或许我代码理解问题，求指教。

- 关于邻居：预训练阶段自己算自己的，不涉及邻居。正式训练时，代码中将自己本身的数据与20个邻居的数据放在一起，形成一个三维矩阵，第一维代表样本个数，第二维代表数据的维度，第三维代表自己+邻居的ID。例如：image_train[:,:,0]代表数据本身，image_train[:,:,1]代表数据的第一个邻居。这样将数据本身与邻居整合在一起，整体作为输入数据，送进网络进行训练。这种设计相当巧妙，相当于以空间换时间，减少训练过程中用来查找邻居的时间。同时，DGG总体损失函数$\underset{\phi ,\theta }{\mathop{\max }}\,\frac{1}{2}\sum\limits_{i=1}^{N}{\sum\limits_{j=1}^{N}{{{w}_{ij}}(L(\theta ,\phi ;{{x}_{i}})+L(\theta ,\phi ;{{x}_{i}},{{x}_{j}}))}}$可以将这两项合并成一项，既计算了自己与自己的损失函数，也计算了自己与邻居的损失函数。

- 关于${{\pi }_{ik}}$：这篇文章中参数${{\pi }_{ik}}$与GMM中的混合比例不太一样，计算${{\pi }_{ik}}$时是通过拉格朗日乘数法进行求解的${{\pi }_{ik}}=\frac{\sum\limits_{j\in {{\Omega }_{i}}}{{{w}_{ij}}{{\gamma }_{ik}}}}{\sum\limits_{j\in {{\Omega }_{i}}}{{{w}_{ij}}\left({{\gamma }_{ik}}+{{\gamma }_{jk}} \right)}}$，但在代码中定义的${{\pi }_{ik}}$的更新公式与原文提到的更新公式不一致。

```
x_mean,_ = vae.get_latent(inputs)
pc = classifer(x_mean).data

pc = torch.mul(pc,weight.view(-1,1))
pc_temp = 0
for idx in range(N_n):
        pc_temp = pc_temp + pc[idx*N_samples:(idx+1)*N_samples,:]
pc_temp1 = []
for idx  in range(N_n):
        pc_temp1.append(pc_temp)

pc = torch.cat(pc_temp1,dim=0)
```

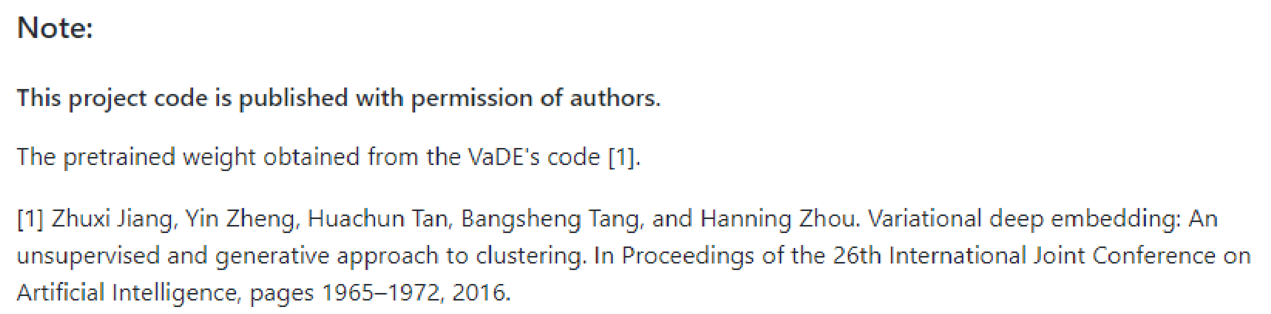
- 同时，在定义GMM类的时候，计算后验概率${{\gamma }_{ik}}$时并没有出现混合比例${{\pi }_{k}}$，没有用GMM中${{\gamma }_{ik}}$的更新公式${{\gamma }_{ik}}=\frac{{{\pi }_{k}}N({{x}_{i}}|{{\mu }_{k}},{{\Sigma }_{k}})}{\sum\limits_{k=1}^{K}{{{\pi }_{k}}N({{x}_{i}}|{{\mu }_{k}},{{\Sigma }_{k}})}}$进行计算。

```python
class GMM_Model(nn.Module):
        def __init__(self,N,K,mean=None,var=None,prior=None):
                super(GMM_Model,self).__init__()
                if mean is not None:
                        self.mean = nn.Parameter(torch.from_numpy(mean).view(1,N,K))
                        self.std = nn.Parameter(torch.sqrt(torch.from_numpy(var)).view(1,N,K))
                else:
                        self.mean = nn.Parameter(torch.randn(1,N,K))
                        self.std = nn.Parameter(torch.ones(1,N,K))

                self.N = N
                self.K = K

        def get_para(self):

                return self.mean, self.std

        def log_prob(self,data_mean,data_logvar,cond_prob,weight):
                term1 = torch.sum(-torch.log((self.std**2)*2*math.pi),dim=1)*0.5
                term2 = torch.sum(-torch.div(torch.pow(data_mean.view(-1,self.N,1)-self.mean,2)+torch.exp(data_logvar).view(-1,self.N,1),self.std**2),dim=1)*0.5
                prob = term2 + term1
                log_p1 = torch.sum(torch.mul(prob,cond_prob),dim=-1)
                log_p = torch.sum(torch.mul(log_p1,weight))

                return log_p

        def compute_prob(self,data):
                prob = torch.exp(torch.sum(-torch.log((self.std**2)*2*math.pi)-torch.div(torch.pow(data.view(-1,self.N,1)-self.mean,2),self.std**2),dim=1)*0.5)
                pc = torch.div(prob,(torch.sum(prob,dim=-1)).view(-1,1)+1e-10)
                return pc
```

- 代码中计算相似度矩阵${w}_{ij}$时，窗口大小sigma的选取有歧义，函数中已经有了输入变量sigma，但是里面又重新定义了sigma=dist_temp[11]，这样的话输入sigma还有什么意义？无论输入多少都无所谓，因为会被新定义的覆盖掉。同时，sigma为什么要这样定义？为什么是[11]？

```python
def compute_weight(inputs,sigma,similarity_type='Gauss',use_gpu=torch.cuda.is_available()):
        dist = torch.sum(torch.pow(inputs-inputs[:,:,0].unsqueeze(2),2),dim=1)
        dist_temp,_ = torch.sort(dist)
        sigma = dist_temp[11]
        dist = dist/sigma
        if similarity_type == 'Gauss':
                Gauss_simi = torch.exp(-dist)
                Gauss_simi[:,0] = torch.sum(Gauss_simi[:,1:],dim=1)
                # simi = torch.div(Gauss_simi,torch.sum(Gauss_simi,dim=1,keepdim=True))
                simi = Gauss_simi
```

- DGG原作者给的代码里面说预训练的参数是从VaDE代码里面获得的。这里VaDE与DGG在训练同一组数据用的VAE网络架构是一致的，因此可以直接拿来用。如果数据是新的，首先需要训练Siamese网络来找数据点的邻居，然后自己构建深度自编码器或者变分自编码器预训练模型参数。



Note:

This project code is published with permission of authors.

The pretrained weight obtained from the VaDE's code [1].

[1] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, pages 1965–1972, 2016.

6. 参考文献

[1] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang, "Deep Clustering by Gaussian Mixture Variational Autoencoders with Graph Embedding", In ICCV 2019.

[2] 论文补充材料：Deep Clustering by Gaussian Mixture Variational Autoencoders with Graph Embedding - Supplementary

[3] DGG Python代码：https://github.com/ngoc-nguyen-0/DGG

[4] 变分深度嵌入(Variational Deep Embedding, VaDE) - 凯鲁嘎吉 - 博客园

[5] 变分推断与变分自编码器 - 凯鲁嘎吉 - 博客园