1. Uniform convergence and Model Selection

(a)由Hoeffding不等式，我们可得对每个$\hat{h}_i$，

$$P(|\epsilon(\hat{h}_i) - \hat{\epsilon}_{S_{\mathrm{cv}}}(\hat{h}_i)| > \gamma) \le 2ke^{-2\gamma^2\beta m}$$
$$P(|\epsilon(\hat{h}_i) - \hat{\epsilon}_{S_{\mathrm{cv}}}(\hat{h}_i)| \le \gamma) \ge 1 - 2ke^{-2\gamma^2\beta m}$$

令

$$2ke^{-2\gamma^2\beta m} = \frac{\delta}{2}$$

可得

$$e^{2\gamma^2\beta m} = \frac{4k}{\delta}$$
$$2\gamma^2\beta m = \log\frac{4k}{\delta}$$
$$\gamma = \sqrt{\frac{1}{2\beta m}\log\frac{4k}{\delta}}$$

因此至少有$1-\frac{\delta}{2}$的概率，对每个$\hat{h}_i$，我们有

$$|\epsilon(\hat{h}_i) - \hat{\epsilon}_{S_{\mathrm{scv}}}(\hat{h}_i)| \le \sqrt{\frac{1}{2\beta m}\log\frac{4k}{\delta}}$$

(b)由(a)可得，$\forall i$

$$\epsilon(\hat{h}_i) \le \hat{\epsilon}_{S_{\mathrm{scv}}}(\hat{h}_i) + \sqrt{\frac{1}{2\beta m}\log\frac{4k}{\delta}} \tag{1}$$
$$\hat{\epsilon}_{S_{\mathrm{scv}}}(\hat{h}_i) \le \epsilon(\hat{h}_i) + \sqrt{\frac{1}{2\beta m}\log\frac{4k}{\delta}} \tag{2}$$

由不等式(2)可得，$\forall i$

$$\begin{aligned}\hat{\epsilon}_{S_{\mathrm{scv}}}(\hat{h}) &= \min_{h\in\{\hat{h}_1,\ldots,\hat{h}_k\}} \hat{\epsilon}_{S_{\mathrm{scv}}}(h) \\ &\le \hat{\epsilon}_{S_{\mathrm{scv}}}(\hat{h}_i) \\ &\le \epsilon(\hat{h}_i) + \sqrt{\frac{1}{2\beta m}\log\frac{4k}{\delta}} \end{aligned} \tag{3}$$

所以，$\forall i$

$$
\begin{aligned}
\epsilon(\hat{h}) &\le \hat{\epsilon}_{S_{\text{scv}}}(\hat{h})+\sqrt{\frac{1}{2\beta m}\log\frac{4k}{\delta}}\\
&\le \epsilon(\hat{h}_i)+\sqrt{\frac{1}{2\beta m}\log\frac{4k}{\delta}}+\sqrt{\frac{1}{2\beta m}\log\frac{4k}{\delta}}\\
&\le \epsilon(\hat{h}_i)+\sqrt{\frac{2}{\beta m}\log\frac{4k}{\delta}}
\end{aligned}
$$

因此

$$
\epsilon(\hat{h}) \le \min_{i=1,\ldots,k}\epsilon(\hat{h}_i)+\sqrt{\frac{2}{\beta m}\log\frac{4k}{\delta}}
$$

(c)由(b)可知,

$$
\epsilon(\hat{h}) \le \epsilon(\hat{h}_j)+\sqrt{\frac{2}{\beta m}\log\frac{4k}{\delta}}
$$

记上述事件为A，记如下事件为B

$$
|\epsilon(\hat{h}_j)-\hat{\epsilon}_{S_{\text{train}}}(h_j^*)| \le \sqrt{\frac{2}{(1-\beta)m}\log\frac{4|\mathcal{H}_j|}{\delta}}, \forall j\in\mathcal{H}_j
$$

因为

$$
P(A)\ge 1-\frac{\delta}{2}, P(B)\ge 1-\frac{\delta}{2}
$$

所以

$$
\begin{aligned}
P(AB) &= P(A)+P(B)-P(A\cup B)\\
&\ge 1-\frac{\delta}{2}+1-\frac{\delta}{2}-1\\
&= 1-\delta
\end{aligned}
$$

注意AB即为如下事件

$$
\epsilon(\hat{h}) \le \epsilon(\hat{h}_j)+\sqrt{\frac{2}{\beta m}\log\frac{4k}{\delta}} \le \left(\hat{\epsilon}_{S_{\text{train}}}(h_j^*)+\sqrt{\frac{2}{(1-\beta)m}\log\frac{4|\mathcal{H}_j|}{\delta}}\right)+\sqrt{\frac{2}{\beta m}\log\frac{4k}{\delta}}, \forall j\in\mathcal{H}_j
$$

$$
\epsilon(\hat{h}) \le \min_{j=1,\ldots,k}\left(\hat{\epsilon}_{S_{\text{train}}}(h_j^*)+\sqrt{\frac{2}{(1-\beta)m}\log\frac{4|\mathcal{H}_j|}{\delta}}\right)+\sqrt{\frac{2}{\beta m}\log\frac{4k}{\delta}}
$$

因此

$$\epsilon(\hat{h}) \le \min_{i=1,\ldots,k}\left(\hat{\epsilon}_{S_{\text{train}}}(h_j^*) + \sqrt{\frac{2}{(1-\beta)m}\log\frac{4|\mathcal{H}_j|}{\delta}}\right) + \sqrt{\frac{2}{\beta m}\log\frac{4k}{\delta}}$$

发生的概率大于等于$1-\delta$

2. VC Dimension

后面都用d表示VC维。

$h(x)=1\{a<x\}$的VC维是1，首先存在1个点可以shatter，其次对于任意两个点$x_1<x_2$，标签$(1,0)$无法表出，所以

$$d=1$$

$h(x)=1\{a<x<b\}$的VC维是2，首先存在2个点可以shatter（作图即可），其次对于任意三个点$x_1<x_2<x_3$，标签$(1,0,1)$无法表出，所以

$$d=2$$

$h(x)=1\{a\sin x>0\}$的VC维是1，首先存在1个点可以shatter（作图即可），其次对于任意两个点$x_1<x_2$，我们有

$$(a\sin x_1)(a\sin x_2)=\sin x_1\sin x_2$$

所以如果

$$\sin x_1\sin x_2>0$$

那么$(1,0),(0,1)$的标签无法表出。如果

$$\sin x_1\sin x_2\le 0$$

那么$(1,1)$的标签无法表出。所以

$$d=1$$

$h(x)=1\{\sin(x+a)>0\}$的VC维是2，首先$0,\frac{\pi}{2}$可以被shatter，其次由于sin函数的周期为2π，我们可以假设所有的点属于$[0,2\pi)$。在继续讨论之前，首先将模型转换，将$[0,2\pi)$上的点映射到单位圆周上，a对应过原点的一条直线，直线一侧的点为1，另一侧为0。

现在假设存在三个点$0\le x_1<x_2<x_3<2\pi$可以被shatter，那么标签$(1,1,1)$表出，由之前讨论可知这表示x_1,x_2,x_3在直线的同侧，那么可以发现$(1,0,1)$必然无法表出，这是因为x_1,x_3在直线的同侧，介于两点之间的点x_2必然属于同侧，所以无法表出$(1,0,1)$

3. ℓ_1 regularization for least squares

(a)首先对$J(\theta)$进行化简：

$$
\begin{aligned}
J(\theta) &= \frac{1}{2}(X\bar{\theta} + X_i\theta_i - \vec{y})^T(X\bar{\theta} + X_i\theta_i - \vec{y}) + \lambda||\bar{\theta}||_1 + \lambda s_i\theta_i \\
&= \frac{1}{2}(X\bar{\theta} - \vec{y})^T(X\bar{\theta} - \vec{y}) + \frac{1}{2}\theta_i^T X_i^T X_i\theta_i + (X\bar{\theta} - \vec{y})^T X_i\theta_i + \lambda||\bar{\theta}||_1 + \lambda s_i\theta_i \\
&= \frac{1}{2}(X\bar{\theta} - \vec{y})^T(X\bar{\theta} - \vec{y}) + \frac{1}{2}X_i^T X_i\theta_i^2 + (X\bar{\theta} - \vec{y})^T X_i\theta_i + \lambda||\bar{\theta}||_1 + \lambda s_i\theta_i
\end{aligned}
$$

所以

$$
\frac{\partial J(\theta)}{\partial \theta_i} = X_i^T X_i\theta_i + (X\bar{\theta} - \vec{y})^T X_i + \lambda s_i
$$

令上式为0可得

$$
\theta_i = -\frac{(X\bar{\theta} - \vec{y})^T X_i + \lambda s_i}{X_i^T X_i}
$$

如果$s_i = 1$，那么$\theta_i \ge 0$，所以我们有

$$
\theta_i = \max\{-\frac{(X\bar{\theta} - \vec{y})^T X_i + \lambda}{X_i^T X_i}, 0\}
$$

如果$s_i = -1$，那么$\theta_i < 0$，所以我们有

$$
\theta_i = \min\{-\frac{(X\bar{\theta} - \vec{y})^T X_i - \lambda}{X_i^T X_i}, 0\}
$$

(b)代码实现的过程中要分别对$s_i = \pm 1$讨论，根据选择产生较小$J(\theta)$对应的θ_i。

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar  9 14:27:42 2019

@author: qinzhen
"""

import numpy as np

X = np.genfromtxt("x.dat")
y = np.genfromtxt("y.dat")
theta_true = np.genfromtxt("theta.dat")

def l1l2(X, y, Lambda):
    #数据维度
    n, d = X.shape
    #设置阈值
    D = 1e-5
    #设置初始值
    theta = np.zeros(d)
    #记录上一轮迭代的theta
    theta_pre = np.copy(theta)
    while True:
```

```
        #坐标下降
        for i in range(d):
            #第i列
            Xi = X[:, i]
            #theta第i个元素为0
            theta[i] = 0
            #计算
            temp1 = X.dot(theta) - y
            temp2 = np.max([- (temp1.T.dot(Xi) + Lambda) / (Xi.T.dot(Xi)), 0])
            temp3 = np.min([- (temp1.T.dot(Xi) - Lambda) / (Xi.T.dot(Xi)), 0])
            #情形1
            theta[i] = temp2
            loss1 = 1 / 2 * np.sum((X.dot(theta) - y) ** 2) + Lambda *
np.sum(np.abs(theta))
            #情形2
            theta[i] = temp3
            loss2 = 1 / 2 * np.sum((X.dot(theta) - y) ** 2) + Lambda *
np.sum(np.abs(theta))

            #根据较小的loss对应的值更新
            if(loss1 < loss2):
                theta[i] = temp2
            else:
                theta[i] = temp3

        #计算误差
        delta = np.linalg.norm(theta - theta_pre)
        if delta < D:
            break

        theta_pre = np.copy(theta)

    return theta
```

(b)对Lambda = 1运行，得到如下结果：

```
theta = L1L2(X, y, 1)
print(theta)
```

```
[ 0.49875676  0.65610562 -0.79057895 -0.6556427  -0.89191611  0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
```

```
 0.          0.          0.          0.          0.          0.
 0.          0.          0.          0.          0.          0.
 0.          0.          0.          0.          0.          0.
 0.          0.          0.          0.        ]
```

可以看到最终的θ是稀疏的，所以可以用l_1正则化进行特征选择，保留系数不为0的特征。

4. K-Means Clustering

这里使用向量化的方法计算每个点距离聚类中心的距离，提高计算效率，介绍如下

假设

$$X=\begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix}\in\mathbb{R}^{m\times d}, Y=\begin{bmatrix} —(y^{(1)})^T— \\ —(y^{(2)})^T— \\ \vdots \\ —(y^{(n)})^T— \end{bmatrix}\in\mathbb{R}^{n\times d}$$

其中$x^{(i)},y^{(i)}\in\mathbb{R}^d$，现在的问题是如何高效计算矩阵$D\in\mathbb{R}^{m\times n}$，其中

$$D_{i,j}=||x^{(i)}-y^{(j)}||^2$$

首先对$D_{i,j}$进行处理

$$\begin{aligned} D_{i,j}&=||x^{(i)}-y^{(j)}||^2 \\ &=(x^{(i)}-y^{(j)})^T(x^{(i)}-y^{(j)}) \\ &=(x^{(i)})^Tx^{(i)}-2(x^{(i)})^Ty^{(j)}+(y^{(j)})^Ty^{(j)} \end{aligned}$$

那么

$$\begin{aligned} D&=\begin{bmatrix} D_{1,1} & \ldots & D_{1,n} \\ \ldots & \ldots & \ldots \\ D_{m,1} & \ldots & D_{m,n} \end{bmatrix} \\ &=\begin{bmatrix} (x^{(1)})^Tx^{(1)}-2(x^{(1)})^Ty^{(1)}+(y^{(1)})^Ty^{(1)} & \ldots & (x^{(1)})^Tx^{(1)}-2(x^{(1)})^Ty^{(n)}+(y^{(n)})^Ty^{(n)} \\ \ldots & \ldots & \ldots \\ (x^{(m)})^Tx^{(m)}-2(x^{(m)})^Ty^{(1)}+(y^{(1)})^Ty^{(1)} & \ldots & (x^{(m)})^Tx^{(m)}-2(x^{(m)})^Ty^{(n)}+(y^{(n)})^Ty^{(n)} \end{bmatrix} \\ &=\begin{bmatrix} (x^{(1)})^Tx^{(1)} & \ldots & (x^{(1)})^Tx^{(1)} \\ \ldots & \ldots & \ldots \\ (x^{(m)})^Tx^{(m)} & \ldots & (x^{(m)})^Tx^{(m)} \end{bmatrix}+\begin{bmatrix} (y^{(1)})^Ty^{(1)} & \ldots & (y^{(n)})^Ty^{(n)} \\ \ldots & \ldots & \ldots \\ (y^{(1)})^Ty^{(1)} & \ldots & (y^{(n)})^Ty^{(n)} \end{bmatrix}-2\begin{bmatrix} (x^{(1)})^Ty^{(1)} & \ldots & (x^{(1)})^Ty^{(n)} \\ \ldots & \ldots & \ldots \\ (x^{(m)})^Ty^{(1)} & \ldots & (x^{(m)})^Ty^{(n)} \end{bmatrix} \\ &=\begin{bmatrix} (x^{(1)})^Tx^{(1)} \\ \ldots \\ (x^{(m)})^Tx^{(m)} \end{bmatrix}\underbrace{\begin{bmatrix} 1 & \ldots & 1 \end{bmatrix}}_{1\times n\text{矩阵}}+\underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{m\times 1\text{矩阵}}\begin{bmatrix} (y^{(1)})^Ty^{(1)} & \ldots & (y^{(n)})^Ty^{(n)} \end{bmatrix}-2XY^T \end{aligned}$$

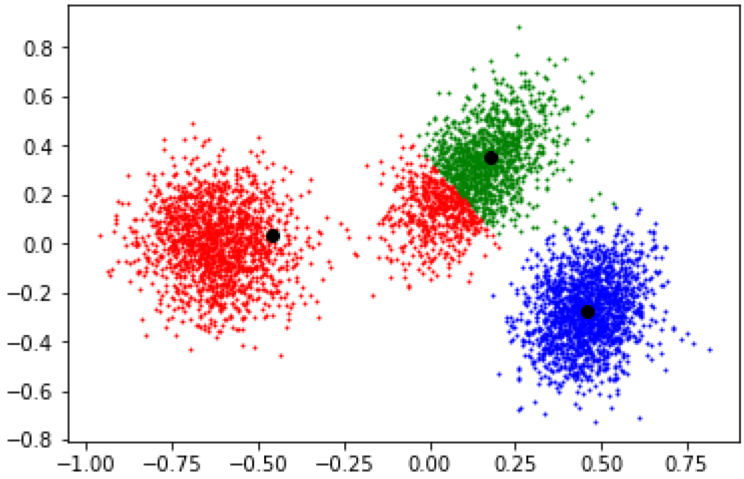
利用numpy的广播机制上式可以简写如下：

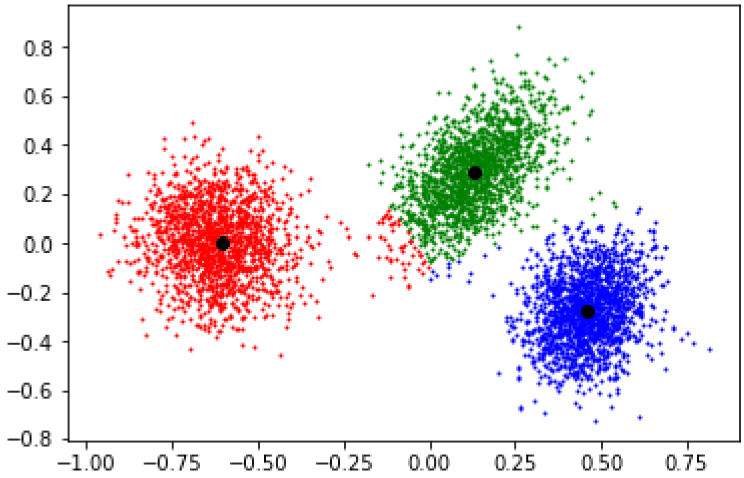
```
#计算距离矩阵
d1 = np.sum(X ** 2, axis=1).reshape(-1, 1)
d2 = np.sum(centroids ** 2, axis=1).reshape(1, -1)

dist = d1 + d2 - 2 * X.dot(centroids.T)
```

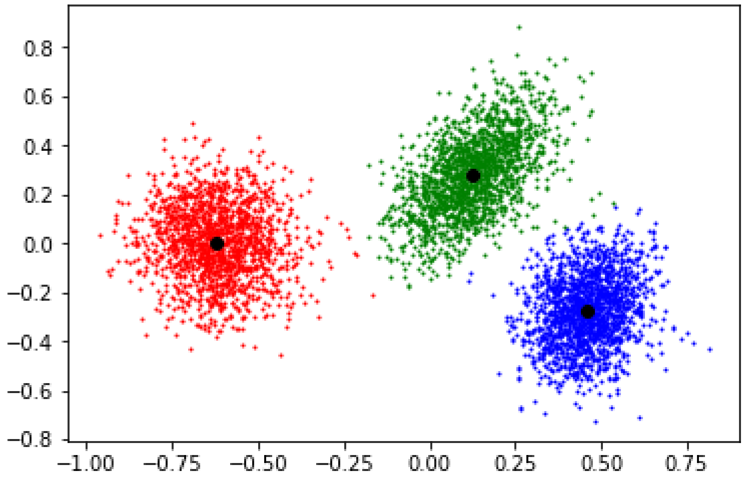
全部代码如下：

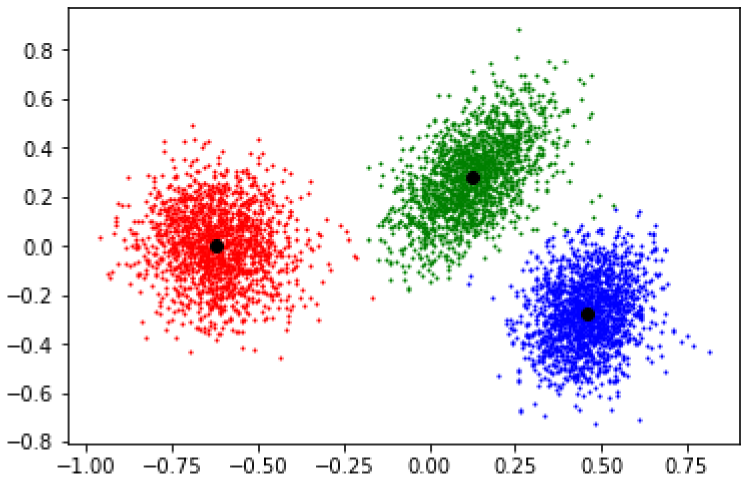
```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar  9 15:41:53 2019

@author: qinzhen
"""

import numpy as np
import matplotlib.pyplot as plt

def draw_clusters(X, clusters, centroids):
    #颜色列表
    c = ["b", "g", "r", "c", "m", "y"]
    #聚类数量
    d = np.max(clusters)
    #画出每种聚类
    for i in range(d+1):
        plt.scatter(X[clusters==i][:, 0], X[clusters==i][:, 1], c=c[i], s=1)

    #画出中心
    plt.scatter(centroids[:, 0], centroids[:, 1], c="black")
    plt.show()

def k_means(X, k, plot=0):
    #数据维度
    n, d = X.shape
    #聚类标签
    clusters = np.zeros(n, dtype=int)
    #初始中心点
    index = np.random.randint(0, n, k)
    #centroids = np.random.rand(k, d)
    centroids = X[index]
    #记录上一轮迭代的聚类中心
    centroids_pre = np.copy(centroids)
    #设置阈值
    D = 1e-5

    while True:
        #计算距离矩阵
        d1 = np.sum(X ** 2, axis=1).reshape(-1, 1)
        d2 = np.sum(centroids ** 2, axis=1).reshape(1, -1)
        dist = d1 + d2 - 2 * X.dot(centroids.T)

        #STEP1:找到最近的中心
```

```
        clusters = np.argmin(dist, axis=1)
        #STEP2:重新计算中心
        for i in range(k):
            centroids[i] = np.mean(X[clusters==i], axis=0)

        #计算误差
        delta = np.linalg.norm(centroids - centroids_pre)

        #判断是否作图
        if plot:
            draw_clusters(X, clusters, centroids)

        if delta < D:
            break

        centroids_pre = np.copy(centroids)

X = np.genfromtxt("X.dat")

k_means(X, 3, plot=1)
```









5. The Generalized EM algorithm

(a)

$$\begin{aligned} l(\theta^{(t+1)}) &\geq \sum_i \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t+1)})}{Q_i^{(t)}(z^{(i)})} \\ &\geq \sum_i \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t)})}{Q_i^{(t)}(z^{(i)})} \\ &= l(\theta^{(t)}) \end{aligned}$$

第一个不等号成立是因为如下不等式对任意Q_i和θ都成立

$$l(\theta) \geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

第二个不等号成立是因为梯度上升，等号成立是由$\theta^{(t)}$的定义。

(b)利用定义求梯度即可

$$
\begin{aligned}
\nabla_\theta \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta) &= \sum_i \frac{\nabla_\theta \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta)}{\sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta)} \\
&= \sum_i \sum_{z^{(i)}} \frac{\nabla_\theta p(x^{(i)}, z^{(i)}; \theta)}{p(x^{(i)}; \theta)}
\end{aligned}
$$

注意到GEM中的梯度为

$$
\begin{aligned}
\nabla_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} &= \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \nabla_\theta \log p(x^{(i)}, z^{(i)}; \theta) \\
&= \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{\nabla_\theta p(x^{(i)}, z^{(i)}; \theta)}{p(x^{(i)}, z^{(i)}; \theta)}
\end{aligned}
$$

注意我们选择

$$
Q_i(z^{(i)}) = p(z^{(i)} | x^{(i)}; \theta) = \frac{p(x^{(i)}, z^{(i)}; \theta)}{p(x^{(i)}; \theta)}
$$

所以GEM中的梯度为

$$
\begin{aligned}
\nabla_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} &= \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{\nabla_\theta p(x^{(i)}, z^{(i)}; \theta)}{p(x^{(i)}, z^{(i)}; \theta)} \\
&= \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{\nabla_\theta p(x^{(i)}, z^{(i)}; \theta)}{p(x^{(i)}; \theta) Q_i(z^{(i)})} \\
&= \sum_i \sum_{z^{(i)}} \frac{\nabla_\theta p(x^{(i)}, z^{(i)}; \theta)}{p(x^{(i)}; \theta)}
\end{aligned}
$$

所以这两者等价。