1. EM for supervised learning

(a)由定义可得

$$
\begin{aligned}
p(y^{(j)}|x^{(j)}) &= p(y^{(j)}|x^{(j)},z^{(j)})p(z^{(j)}|x^{(j)}) \\
&= \frac{1}{\sqrt{2\pi}\sigma}\exp\Big(-\frac{(y^{(j)}-\theta_{z^{(j)}}^T x^{(j)})^2}{2\sigma^2}\Big)g(\phi^T x^{(j)})^{z^{(j)}}(1-g(\phi^T x^{(j)}))^{1-z^{(j)}}
\end{aligned}
$$

概率似然函数为

$$
L = \prod_{j=1}^m \frac{1}{\sqrt{2\pi}\sigma}\exp\Big(-\frac{(y^{(j)}-\theta_{z^{(j)}}^T x^{(j)})^2}{2\sigma^2}\Big)g(\phi^T x^{(j)})^{z^{(j)}}(1-g(\phi^T x^{(j)}))^{1-z^{(j)}}
$$

对数似然函数为

$$
\begin{aligned}
l &= \log L \\
&= \sum_{j=1}^m \log\Big(\frac{1}{\sqrt{2\pi}\sigma}\exp\Big(-\frac{(y^{(j)}-\theta_{z^{(j)}}^T x^{(j)})^2}{2\sigma^2}\Big)g(\phi^T x^{(j)})^{z^{(j)}}(1-g(\phi^T x^{(j)}))^{1-z^{(j)}}\Big) \\
&= -m\log(\sqrt{2\pi}\sigma) + \sum_{j=1}^m\Big(-\frac{(y^{(j)}-\theta_{z^{(j)}}^T x^{(j)})^2}{2\sigma^2} + z^{(j)}\log(g(\phi^T x^{(j)})) + (1-z^{(j)})\log(1-g(\phi^T x^{(j)}))\Big)
\end{aligned}
$$

关于θ_i求梯度可得

$$
\begin{aligned}
\nabla_{\theta_i} l &= \sum_{j=1}^m -\frac{1}{2\sigma^2}\Big(2(\theta_{z^{(j)}}^T x^{(j)} - y^{(j)})x^{(j)}1\{z^{(j)}=i\}\Big) \\
&= -\frac{1}{\sigma^2}\Big((\theta_{z^{(j)}}^T x^{(j)} - y^{(j)})x^{(j)}1\{z^{(j)}=i\}\Big) \\
&= -\frac{1}{\sigma^2}\Big(\sum_{j=1}^m\big((x^{(j)})^T\theta_{z^{(j)}} - y^{(j)}\big)x^{(j)}1\{z^{(j)}=i\}\Big) \\
&= -\frac{1}{\sigma^2}\Big(\big(\sum_{j=1}^m (x^{(j)})^T x^{(j)}1\{z^{(j)}=i\}\big)\theta_i - \sum_{j=1}^m 1\{z^{(j)}=i\}y^{(j)}x^{(j)}\Big)
\end{aligned}
$$

令上式为0可得

$$
\big(\sum_{j=1}^m (x^{(j)})^T x^{(j)}1\{z^{(j)}=i\}\big)\theta_i - \sum_{j=1}^m 1\{z^{(j)}=i\}y^{(j)}x^{(j)} = 0
$$

$$
\theta_i = \big(\sum_{j=1}^m (x^{(j)})^T x^{(j)}1\{z^{(j)}=i\}\big)^{-1}\Big(\sum_{j=1}^m 1\{z^{(j)}=i\}y^{(j)}x^{(j)}\Big)
$$

关于ϕ求梯度可得

$$
\begin{aligned}
\nabla_\phi l &= \sum_{j=1}^m \Big(z^{(j)} \frac{1}{g(\phi^T x^{(j)})} g(\phi^T x^{(j)})(1-g(\phi^T x^{(j)}))x^{(j)} - (1-z^{(j)}) \frac{1}{1-g(\phi^T x^{(j)})} g(\phi^T x^{(j)})(1-g(\phi^T x^{(j)}))x^{(j)} \Big) \\
&= \sum_{j=1}^m \Big(z^{(j)}(1-g(\phi^T x^{(j)}))x^{(j)} - (1-z^{(j)}) g(\phi^T x^{(j)})x^{(j)} \Big) \\
&= \sum_{j=1}^m \big(z^{(j)} - g(\phi^T x^{(j)}) \big) x^{(j)}
\end{aligned}
$$

接着求Hessian矩阵，注意上述梯度第s个分量为$\sum_{j=1}^m \big(z^{(j)} - g(\phi^T x^{(j)}) \big) x_s^{(j)}$，求二阶偏导数：

$$
\begin{aligned}
\frac{\partial(\sum_{j=1}^m \big(z^{(j)} - g(\phi^T x^{(j)}) \big) x_s^{(j)})}{\partial \phi_t} &= \sum_{j=1}^m -x_s^{(j)} \frac{\partial(g(\phi^T x^{(j)}))}{\partial \phi_t} \\
&= \sum_{j=1}^m -x_s^{(j)} g(\phi^T x^{(j)})(1-g(\phi^T x^{(j)}))x_t^{(j)}
\end{aligned}
$$

记

$$
X = \begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix}, \vec z = \begin{bmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(m)} \end{bmatrix}
$$

$$
\vec g = \begin{bmatrix} g(\phi^T x^{(1)}) \\ g(\phi^T x^{(2)}) \\ \vdots \\ g(\phi^T x^{(m)}) \end{bmatrix}, D = \mathrm{diag}\{ g(\phi^T x^{(1)})(1-g(\phi^T x^{(1)})), \ldots, g(\phi^T x^{(m)})(1-g(\phi^T x^{(m)})) \}
$$

那么关于ϕ的梯度为

$$
X^T(\vec z - \vec g)
$$

Hessian矩阵为

$$
-X^T D X
$$

(b)

$$
\begin{aligned}
p(y^{(j)}|x^{(j)}) &= \sum_{z^{(j)}} p(y^{(j)}|x^{(j)}, z^{(j)}) p(z^{(j)}|x^{(j)}) \\
&= \sum_{z^{(j)}} \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(-\frac{(y^{(j)} - \theta_{z^{(j)}}^T x^{(j)})^2}{2\sigma^2} \Big) g(\phi^T x^{(j)})^{z^{(j)}} (1-g(\phi^T x^{(j)}))^{1-z^{(j)}}
\end{aligned}
$$

概率似然函数为

$$L=\prod_{j=1}^m\sum_{z^{(j)}}\frac{1}{\sqrt{2\pi}\sigma}\exp\Big(-\frac{(y^{(j)}-\theta_{z^{(j)}}^Tx^{(j)})^2}{2\sigma^2}\Big)g(\phi^Tx^{(j)})^{z^{(j)}}(1-g(\phi^Tx^{(j)}))^{1-z^{(j)}}$$

对数似然函数为

$$\begin{aligned}l&=\sum_{j=1}^m\log\Big(\sum_{z^{(j)}}\frac{1}{\sqrt{2\pi}\sigma}\exp\Big(-\frac{(y^{(j)}-\theta_{z^{(j)}}^Tx^{(j)})^2}{2\sigma^2}\Big)g(\phi^Tx^{(j)})^{z^{(j)}}(1-g(\phi^Tx^{(j)}))^{1-z^{(j)}}\Big)\\&=\sum_{j=1}^m\log\Big(\sum_{z^{(j)}}p(y^{(j)}|x^{(j)},z^{(j)})p(z^{(j)}|x^{(j)})\Big)\end{aligned}$$

为了使用EM算法，我们首先求后验概率

$$Q(z^{(j)})=p(z^{(j)}|x^{(j)},y^{(j)})$$

事实上，我们有

$$\begin{aligned}Q(z^{(j)})&=p(z^{(j)}|x^{(j)},y^{(j)})\\&=\frac{p(x^{(j)},y^{(j)},z^{(j)})}{p(x^{(j)},y^{(j)})}\\&=\frac{p(y^{(j)}|x^{(j)},z^{(j)})p(z^{(j)}|x^{(j)})}{\sum_z p(y^{(j)}|x^{(j)},z)p(z|x^{(j)})}\end{aligned}$$

上述每一项都可以根据定义计算得到。

为了方便叙述，记

$$w_k^{(j)}=Q(z^{(j)}=k)$$

那么由EM算法，接下来我们需要最大化

$$\sum_{j=1}^m\sum_k w_k^{(j)}\log\Big(\frac{p(y^{(j)}|x^{(j)},z^{(j)})p(z^{(j)}|x^{(j)})}{w_k^{(j)}}\Big)=\sum_{j=1}^m\sum_k w_k^{(j)}\Big(\log\big(p(y^{(j)}|x^{(j)},z^{(j)}=k)\big)+\log\big(p(z^{(j)}=k|x^{(j)})\big)-\log w_k^{(j)}\Big)$$

记上式为l_1，关于θ_i求梯度可得

$$
\begin{aligned}
\nabla_{\theta_i} l_1 &= \nabla_{\theta_i} \sum_{j=1}^{m} \sum_{k} w_k^{(j)} \Big(\log\big(p(y^{(j)}|x^{(j)}, z^{(j)} = k)\big) + \log\big(p(z^{(j)} = k|x^{(j)})\big) - \log w_k^{(j)} \Big) \\
&= \nabla_{\theta_i} \sum_{j=1}^{m} \sum_{k} w_k^{(j)} \log\big(p(y^{(j)}|x^{(j)}, z^{(j)} = k)\big) \\
&= \nabla_{\theta_i} \sum_{j=1}^{m} \sum_{k} w_k^{(j)} \Big(\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{(y^{(j)} - \theta_k^T x^{(j)})^2}{2\sigma^2} \Big) \\
&= \sum_{j=1}^{m} \sum_{k} w_k^{(j)} (-\frac{1}{2\sigma^2}) \Big(2(\theta_k^T x^{(j)} - y^{(j)}) x^{(j)} \Big) 1\{k = i\} \\
&= -\frac{1}{\sigma^2} \sum_{j=1}^{m} \sum_{k} w_k^{(j)} (\theta_i^T x^{(j)} - y^{(j)}) x^{(j)} 1\{k = i\} \\
&= -\frac{1}{\sigma^2} \sum_{j=1}^{m} \sum_{z^{(j)}} w_{z^{(j)}}^{(j)} (\theta_i^T x^{(j)} - y^{(j)}) x^{(j)} 1\{z^{(j)} = i\}
\end{aligned}
$$

其中最后一步是因为$z^{(j)} = k$。令上式为0可得

$$
\begin{aligned}
(\sum_{j=1}^{m} \sum_{z^{(j)}} (x^{(j)})^T x^{(j)} w_{z^{(j)}}^{(j)} 1\{z^{(j)} = i\}) \theta_i - \sum_{j=1}^{m} \sum_{k} w_{z^{(j)}}^{(j)} 1\{z^{(j)} = i\} y^{(j)} x^{(j)} = 0 \\
\theta_i = (\sum_{j=1}^{m} \sum_{z^{(j)}} (x^{(j)})^T x^{(j)} w_{z^{(j)}}^{(j)} 1\{z^{(j)} = i\})^{-1} \Big(\sum_{j=1}^{m} \sum_{z^{(j)}} w_{z^{(j)}}^{(j)} 1\{z^{(j)} = i\} y^{(j)} x^{(j)} \Big)
\end{aligned}
$$

关于ϕ求梯度可得

$$
\begin{aligned}
\nabla_{\phi} l_1 &= \nabla_{\phi} \sum_{j=1}^{m} \sum_{k} w_k^{(j)} \Big(\log\big(p(y^{(j)}|x^{(j)}, z^{(j)} = k)\big) + \log\big(p(z^{(j)} = k|x^{(j)})\big) - \log w_k^{(j)} \Big) \\
&= \nabla_{\phi} \sum_{j=1}^{m} \sum_{k} w_k^{(j)} \Big(\log\big(p(z^{(j)} = k|x^{(j)})\big) \Big) \\
&= \nabla_{\phi} \sum_{j=1}^{m} \sum_{k} w_k^{(j)} \Big(\log\big(g(\phi^T x^{(j)})^k (1 - g(\phi^T x^{(j)}))^{1-k}\big) \Big) \\
&= \nabla_{\phi} \sum_{j=1}^{m} w_0^{(j)} \log(1 - g(\phi^T x^{(j)})) + w_1^{(j)} \log(g(\phi^T x^{(j)})) \\
&= \sum_{j=1}^{m} -w_0^{(j)} g(\phi^T x^{(j)}) x^{(j)} + w_1^{(j)} (1 - g(\phi^T x^{(j)})) x^{(j)} \\
&= \sum_{j=1}^{m} (w_1^{(j)} - g(\phi^T x^{(j)})) x^{(j)}
\end{aligned}
$$

最后一步利用到了

$$
w_0^{(j)} + w_1^{(j)} = 1
$$

记

$$\vec{w}=\begin{bmatrix} w^{(1)} \\ w^{(2)} \\ \vdots \\ w^{(m)} \end{bmatrix}$$

那么关于ϕ的梯度以及Hessian矩阵和(a)的形式一样：

关于ϕ的梯度为

$$X^T(\vec{w}-\vec{g})$$

Hessian矩阵为

$$-X^TDX$$

2. Factor Analysis and PCA

(a)我们可以等价的定义

$$\begin{aligned} z &\sim \mathcal{N}(0,I) \\ \epsilon &\sim \mathcal{N}(0,\sigma^2 I) \\ x &= Uz+\epsilon \end{aligned}$$

由$z\sim\mathcal{N}(0,I)$我们知道$\mathbb{E}[z]=0$。所以，我们有

$$\begin{aligned} \mathbb{E}[x] &= \mathbb{E}[Uz+\epsilon] \\ &= U\mathbb{E}[z]+\mathbb{E}[\epsilon] \\ &= 0 \end{aligned}$$

将上述结果合并，我们有

$$\mu_{zx}=\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

接着求协方差。因为$z\sim\mathcal{N}(0,I)$，我们可以轻松得到$\Sigma_{zz}=\mathrm{Cov}(z)=I$。并且，我们有

$$\begin{aligned} \mathbb{E}[(z-\mathbb{E}[z])(x-\mathbb{E}[x])^T] &= \mathbb{E}[z(Uz+\epsilon)^T] \\ &= \mathbb{E}[zz^T]U^T+\mathbb{E}[z\epsilon^T] \\ &= U^T \end{aligned}$$

类似的，我们可以按如下方式计算出Σ_{xx}

$$\begin{aligned} \mathbb{E}[(x-\mathbb{E}[x])(x-\mathbb{E}[x])^T] &= \mathbb{E}[(Uz+\epsilon)(Uz+\epsilon)^T] \\ &= \mathbb{E}[Uzz^TU^T+\epsilon z^TU^T+Uz\epsilon^T+\epsilon\epsilon^T] \\ &= U\mathbb{E}[zz^T]U^T+\mathbb{E}[\epsilon\epsilon^T] \\ &= UU^T+\sigma^2 I \end{aligned}$$

$$\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\Big(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & U^T \\ U & UU^T + \sigma^2 I \end{bmatrix} \Big)$$

回顾之前的结论

$$\begin{aligned} &(1) x_1 \sim \mathcal{N}(\mu_1, \Sigma_{11}) \\ &(2) x_1 | x_2 \sim \mathcal{N}(\mu_{1|2}, \Sigma_{1|2}) \\ \text{其中} \ &\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \\ &\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned}$$

所以

$$\begin{aligned} \mu_{z|x} &= 0 + U^T(UU^T + \sigma^2 I)^{-1} x \\ &= U^T(\sigma^2 I + UU^T)^{-1} x \\ &= (\sigma^2 + U^T U)^{-1} U^T x \\ &= \frac{U^T x}{\sigma^2 + U^T U} \\ \Sigma_{z|x} &= 1 - U^T(UU^T + \sigma^2 I)^{-1} U \\ &= 1 - U^T(\sigma^2 I + UU^T)^{-1} U \\ &= 1 - U^T U(\sigma^2 I + U^T U)^{-1} \\ &= \frac{\sigma^2}{\sigma^2 + U^T U} \end{aligned}$$

(b)不难发现x的边际分布为

$$x \sim \mathcal{N}\Big(0, UU^T + \sigma^2 I\Big)$$

所以对数似然函数为

$$l = \log \prod_{i=1}^{m} \frac{1}{(2\pi)^{(n)/2} |UU^T + \sigma^2 I|^{1/2}} \exp\Big(- \frac{1}{2}(x^{(i)})^T (UU^T + \sigma^2 I)^{-1} (x^{(i)}) \Big)$$

接着使用EM算法，E步骤很简单，只要取

$$Q_i(z^{(i)}) = p(z^{(i)} | x^{(i)})$$

由之前讨论，不难发现

$$Q_i(z^{(i)}) = \frac{1}{\sqrt{(2\pi)\frac{\sigma^2}{\sigma^2 + U^T U}}} \exp\Big(- \frac{(z^{(i)} - (\sigma^2 + U^T U)^{-1} U^T x^{(i)})^2}{2\frac{\sigma^2}{\sigma^2 + U^T U}} \Big)$$

接着，我们需要最大化

$$
\begin{aligned}
\sum_{i=1}^{m}\int_{z^{(i)}} Q_i(z^{(i)})\log\frac{p(x^{(i)},z^{(i)})}{Q_i(z^{(i)})}dz^{(i)} &= \sum_{i=1}^{m}\int_{z^{(i)}} Q_i(z^{(i)})\log\frac{p(x^{(i)}|z^{(i)})p(z^{(i)})}{Q_i(z^{(i)})}dz^{(i)} \\
&= \sum_{i=1}^{m}\mathbb{E}_{z^{(i)}\sim Q_i}\big[\log p(x^{(i)}|z^{(i)}) + \log p(z^{(i)}) - \log Q_i(z^{(i)})\big]
\end{aligned}
$$

弃不依赖参数U的项（Q_i虽然包含U，但是在执行更新时是固定的），可以发现我们需要最大化：

$$
\begin{aligned}
&\sum_{i=1}^{m}\mathbb{E}\big[\log p(x^{(i)}|z^{(i)})\big] \\
&= \sum_{i=1}^{m}\mathbb{E}\Big[\log\frac{1}{(2\pi)^{\frac{n}{2}}|\sigma^2 I_n|^{\frac{1}{2}}}\exp\Big(-\frac{1}{2}(x^{(i)}-Uz^{(i)})^T(\sigma^2 I_n)^{-1}(x^{(i)}-Uz^{(i)})\Big)\Big] \\
&= \sum_{i=1}^{m}\mathbb{E}\Big[-\frac{1}{2}\log\sigma^{2n} - \frac{n}{2}\log(2\pi) - \frac{1}{2}(x^{(i)}-Uz^{(i)})^T(\sigma^2 I_n)^{-1}(x^{(i)}-Uz^{(i)})\Big] \\
&= \sum_{i=1}^{m}\mathbb{E}\Big[-\frac{n}{2}\log\sigma^{2} - \frac{n}{2}\log(2\pi) - \frac{1}{2\sigma^2}(x^{(i)}-Uz^{(i)})^T(x^{(i)}-Uz^{(i)})\Big]
\end{aligned}
$$

让我们关于U最大化上式，这里需要利用

$$
\begin{gathered}
\mathrm{trace}(a) = a(a\in\mathbb{R}) \\
\mathrm{trace}(AB) = \mathrm{trace}(BA) \\
\nabla_A \mathrm{trace}(ABA^TC) = CAB + C^TAB \\
\nabla_A \mathrm{trace}(AB) = B^T \\
\nabla_{A^T} f(A) = (\nabla_A f(A))^T
\end{gathered}
$$

注意到只有最后一项依赖U，求梯度可得

$$
\begin{aligned}
&\nabla_U \sum_{i=1}^{m} -\frac{1}{2\sigma^2}\mathbb{E}\Big[(x^{(i)}-Uz^{(i)})^T(x^{(i)}-Uz^{(i)})\Big] \\
&= -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\nabla_U\mathbb{E}\Big[(x^{(i)})^Tx^{(i)} - 2(x^{(i)})^TUz^{(i)} + (z^{(i)})^TU^TUz^{(i)}\Big] \\
&= -\frac{1}{2\sigma^2}\nabla_U\sum_{i=1}^{m}\mathbb{E}\Big[-2\mathrm{trace}\big((x^{(i)})^TUz^{(i)}\big) + \mathrm{trace}\big((z^{(i)})^TU^TUz^{(i)}\big)\Big] \\
&= -\frac{1}{2\sigma^2}\nabla_U\sum_{i=1}^{m}\mathbb{E}\Big[-2\mathrm{trace}\big(Uz^{(i)}(x^{(i)})^T\big) + \mathrm{trace}\big(Uz^{(i)}(z^{(i)})^TU^T\big)\Big] \\
&= -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\mathbb{E}\Big[-2x^{(i)}(z^{(i)})^T + 2Uz^{(i)}(z^{(i)})^T\Big] \\
&= -\frac{1}{\sigma^2}\sum_{i=1}^{m}(U\mathbb{E}[z^{(i)}(z^{(i)})^T] - x^{(i)}\mathbb{E}[(z^{(i)})^T])
\end{aligned}
$$

令上式为0可得

$$
\begin{aligned}
U &= \Big(\sum_{i=1}^m x^{(i)}\mathbb{E}[(z^{(i)})^T]\Big)\Big(\sum_{i=1}^m \mathbb{E}[z^{(i)}(z^{(i)})^T]\Big)^{-1}\\
&= \Big(\sum_{i=1}^m x^{(i)}\mu^T_{z^{(i)}|x^{(i)}}\Big)\Big(\sum_{i=1}^m \Sigma_{z^{(i)}|x^{(i)}} + \mu_{z^{(i)}|x^{(i)}}\mu^T_{z^{(i)}|x^{(i)}}\Big)^{-1}
\end{aligned}
$$

其中

$$
\begin{aligned}
\mu_{z^{(i)}|x^{(i)}} &= \frac{U^Tx^{(i)}}{\sigma^2+U^TU}\\
\Sigma_{z^{(i)}|x^{(i)}} &= \frac{\sigma^2}{\sigma^2+U^TU}
\end{aligned}
$$

(c)如果$\sigma^2\to 0$，那么

$$
\begin{aligned}
\mu_{z^{(i)}|x^{(i)}} &= \frac{U^Tx^{(i)}}{\sigma^2+U^TU} \to \frac{U^Tx^{(i)}}{U^TU}\\
\Sigma_{z^{(i)}|x^{(i)}} &= \frac{\sigma^2}{\sigma^2+U^TU} \to 0
\end{aligned}
$$

如果记

$$
w_i = \mu_{z^{(i)}|x^{(i)}} \to \frac{U^Tx^{(i)}}{U^TU}
$$

所以E步骤可以化为

$$
w = \frac{XU}{U^TU}
$$

M步骤可以化为

$$
\begin{aligned}
U &= \Big(\sum_{i=1}^m x^{(i)}\mu^T_{z^{(i)}|x^{(i)}}\Big)\Big(\sum_{i=1}^m \Sigma_{z^{(i)}|x^{(i)}} + \mu_{z^{(i)}|x^{(i)}}\mu^T_{z^{(i)}|x^{(i)}}\Big)^{-1}\\
&\to \Big(\sum_{i=1}^m x^{(i)}w_i^T\Big)\Big(\sum_{i=1}^m w_iw_i^T\Big)^{-1}\\
&= X^Tw(w^Tw)^{-1}\\
&= \frac{X^Tw}{w^Tw}
\end{aligned}
$$

如果在EM步骤后U不变，那么我们有

$$
U = \frac{X^T\frac{XU}{U^TU}}{(\frac{XU}{U^TU})^T\frac{XU}{U^TU}} = X^TXU\frac{U^TU}{U^TX^TXU}
$$

注意这里$\frac{U^TU}{U^TX^TXU}$为标量，记其为$\frac{1}{\lambda}$，那么

$$
X^TXU = \lambda U
$$

所以结论成立。

3. PCA and ICA for Natural Images

这里要对讲义中ICA的更新公式稍作变形，首先回顾之前的公式：

对于训练样本$x^{(i)}$，随机梯度下降法为

$$W := W + \alpha\left(\begin{bmatrix}1-2g(w_1^Tx^{(i)})\\1-2g(w_2^Tx^{(i)})\\ \ldots\\1-2g(w_n^Tx^{(i)})\end{bmatrix}x^{(i)T} + (W^T)^{-1}\right)$$

记

$$X = \begin{bmatrix}—(x^{(1)})^T—\\—(x^{(2)})^T—\\ \vdots\\—(x^{(m)})^T—\end{bmatrix}, W = \begin{bmatrix}—(w_1)^T—\\—(w_2)^T—\\ \vdots\\—(w_n)^T—\end{bmatrix}$$

那么

$$WX^T = \begin{bmatrix}w_1^Tx^{(1)} & \ldots & w_1^Tx^{(m)}\\ \vdots & \vdots & \vdots\\ w_n^Tx^{(1)} & \ldots & w_n^Tx^{(m)}\end{bmatrix}$$

所以可以利用该矩阵得到

$$G = \begin{bmatrix}1-g(w_1^Tx^{(1)}) & \ldots & 1-g(w_1^Tx^{(m)})\\ \vdots & \vdots & \vdots\\ 1-g(w_n^Tx^{(1)}) & \ldots & 1-g(w_n^Tx^{(m)})\end{bmatrix}$$

对应代码为

```
WX = W.dot(X1.T)
G = 1 - 2 * sigmoid(WX)
```

现在考虑整体梯度

$$\sum_{i=1}^{m}\left(\begin{bmatrix}1-2g(w_1^Tx^{(i)})\\1-2g(w_2^Tx^{(i)})\\\cdots\\1-2g(w_n^Tx^{(i)})\end{bmatrix}{x^{(i)}}^T+(W^T)^{-1}\right)$$

化简可得上式为

$$GX+m(W^T)^{-1}$$

对应代码为:

```
grad = G + batch * inv(W.T)
```

这里我将matlab代码改写为python，下面是代码:

```
import numpy as np
from numpy.linalg import eigh, inv, svd, norm
import matplotlib.pyplot as plt
from matplotlib.image import imread

patch_size = 16

X_ica = np.zeros((patch_size*patch_size, 40000))
idx = 0
for s in range(1, 5):
    #转换为浮点型
    image = imread("./images/{}.jpg".format(s)).astype(np.float32)
    #获得数据维度
    a, b, c = np.shape(image)
    y = a
    x = b * c
    #注意这里order要选择F
    image = image.reshape(y, x, order="F")
    for i in range(1, y//patch_size+1):
        for j in range(1, x//patch_size+1):
            patch = image[(i-1)*patch_size: i*patch_size, (j-1)*patch_size: j*patch_size]
            X_ica[:, idx] = patch.reshape(-1, 1, order="F").flatten()
            idx += 1

X_ica = X_ica[:, 0: idx]

#正定矩阵分解
W = 1 / X_ica.shape[1] * X_ica.dot(X_ica.T)
w, v = eigh(W)
w = np.diag(1 / np.sqrt(w))
W_z = v.dot(w).dot(v.T)

X_ica = X_ica - np.mean(X_ica, axis=1).reshape(-1, 1)
X_pca = X_ica

X_ica = 2 * W_z.dot(X_ica)
```

```
X_pca = X_pca / np.std(X_pca, axis=1).reshape(-1, 1)
```

之前为预处理函数，可以忽略细节。

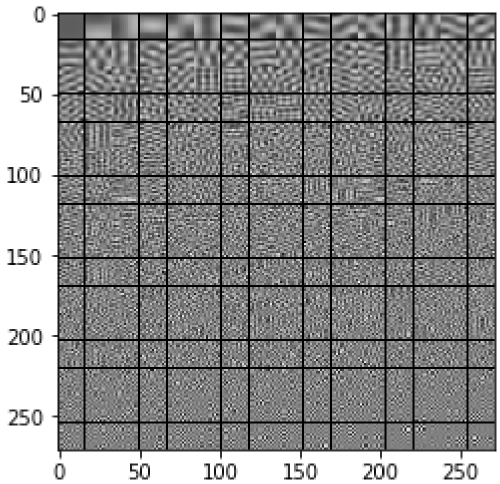
```
#### PCA
def pca(X):
    U, S, V = svd(X.dot(X.T))

    return U

def plot_pca_filters(U):
    n = (patch_size + 1) * patch_size - 1
    big_filters = np.min(U) * np.ones((n, n))
    for i in range(patch_size):
        for j in range(patch_size):
            big_filters[i*(patch_size+1): (i+1)*(patch_size+1)-1, j*(patch_size+1): (j+1)*
(patch_size+1)-1] = U[:, i*patch_size+j].reshape(patch_size, patch_size)

    plt.imshow(big_filters)
    plt.gray()
    plt.show()

U = pca(X_pca)
plot_pca_filters(U)
```




```
#### ICA
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

X = X_ica.copy()

X = X.T
n, d = X.shape
W = np.eye(d)
batch = 100
alpha = 0.0005
eps = 1e-1
```

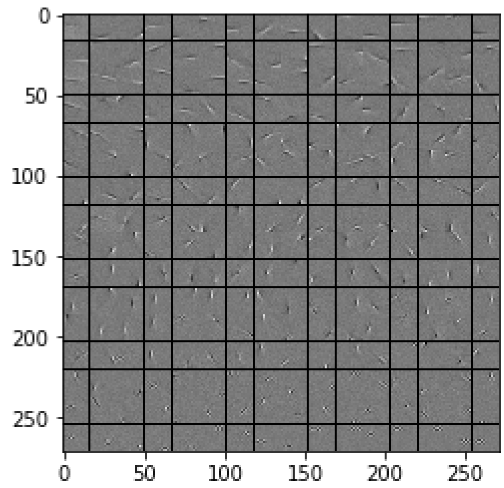
```python
for i in range(10):
    print("第{}轮".format(i+1))
    np.random.permutation(X)
    for j in range(n // batch):
        X1 = X[j * batch: (j+1) * batch, :]
        WX = W.dot(X1.T)
        grad = (1 - 2 * sigmoid(WX)).dot(X1) + batch * inv(W.T)
        W += alpha * grad


def plot_ica_filters(W):
    F = W.dot(W_z)
    norms = norm(F, axis=1)
    idxs = np.argsort(norms)
    norms = np.sort(norms)
    n = (patch_size + 1) * patch_size - 1
    big_filters = np.min(W) * np.ones((n, n))
    for i in range(patch_size):
        for j in range(patch_size):
            temp = W[idxs[i*patch_size+j], :].reshape(patch_size, patch_size)
            big_filters[i*(patch_size+1): (i+1)*(patch_size+1)-1, j*(patch_size+1): (j+1)*
(patch_size+1)-1] = temp

    plt.imshow(big_filters)
    plt.gray()
    plt.show()

plot_ica_filters(W)
```

```
第1轮
第2轮
第3轮
第4轮
第5轮
第6轮
第7轮
第8轮
第9轮
第10轮
```



4. Convergence of Policy Iteration

(a)利用定义即可：

$$
\begin{aligned}
B(V_1)(s) &= V_1'(s) \\
&= R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V_1(s') \\
&\le R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V_2(s') \\
&= V_2'(s) \\
&= B(V_2)(s)
\end{aligned}
$$

(b)注意由定义我们有

$$
\begin{aligned}
B^\pi(V)(s) &= R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V(s') \\
V^\pi(s) &= R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s')
\end{aligned}
$$

所以

$$
\begin{aligned}
|B^\pi(V)(s) - V^\pi(s)| &= |\gamma \sum_{s' \in S} P_{s\pi(s)}(s') V(s') - \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s')| \\
&= \gamma |\sum_{s' \in S} P_{s\pi(s)}(s') (V(s') - V^\pi(s'))| \\
&\le \gamma \sum_{s' \in S} P_{s\pi(s)}(s') |V(s') - V^\pi(s')| \\
&\le \gamma \sum_{s' \in S} P_{s\pi(s)}(s') \|V - V^\pi\|_\infty \\
&= \gamma \|V - V^\pi\|_\infty
\end{aligned}
$$

因此

$$\|B^\pi(V)-V^\pi\|_\infty=\max_{s\in\mathcal{S}}|B^\pi(V)(s)-V^\pi(s)|\le\gamma\|V-V^\pi\|_\infty$$

(c)由π'的定义可得

$$R(s)+\gamma\sum_{s'\in S}P_{s\pi(s)}(s')V^\pi(s')\le R(s)+\gamma\sum_{s'\in S}P_{s\pi'(s)}(s')V^\pi(s')$$

所以

$$\begin{aligned}V^\pi(s)&=R(s)+\gamma\sum_{s'\in S}P_{s\pi(s)}(s')V^\pi(s')\\&\le R(s)+\gamma\sum_{s'\in S}P_{s\pi'(s)}(s')V^\pi(s')\\&=B^{\pi'}(V^\pi)(s)\end{aligned}$$

由(a)可得

$$B^{\pi'}(V^\pi)(s)\le B^{\pi'}(B^{\pi'}(V^\pi))(s)$$

所以

$$V^\pi(s)\le B^{\pi'}(V^\pi)(s)\le B^{\pi'}(B^{\pi'}(V^\pi))(s)$$

对右边不断作用$B^{\pi'}$，然后结合(b)，我们得到

$$V^\pi(s)\le B^{\pi'}(B^{\pi'}(\ldots B^{\pi'}(V)\ldots))=V^{\pi'}(s)$$

(d)因为状态和行动有限，所以

$$V^\pi(s)$$

的取值有限，由(c)可得策略迭代的算法会导致$V^\pi(s)$非降，即

$$V^\pi(s)\le V^{\pi'}(s)$$

所以最终必然会达到

$$\pi'=\pi$$

所以

$$\pi(s)=\arg\max_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}}P_{sa}(s')V^\pi(s')$$

带入$V^\pi(s)$的定义可得

$$V^\pi(s)=R(s)+\gamma\max_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}}P_{sa}(s')V^\pi(s')$$

5. Reinforcement Learning: The Mountain Car

题目的公式有误，正确的如下：

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\Big(R(s') + \gamma \max_{a'\in\mathcal{A}} Q(s',a')\Big)$$

这里没有将代码改写为python，matlab代码参考了标准答案，结果如下：

```
function [q, steps_per_episode] = qlearning(episodes)

% set up parameters and initialize q values
alpha = 0.05;
gamma = 0.99;
num_states = 100;
num_actions = 2;
actions = [-1, 1];
q = zeros(num_states, num_actions);
steps_per_episode = zeros(1, episodes);

for i=1:episodes,
  [x, s, absorb] =  mountain_car([0.0 -pi/6], 0);
  %%% YOUR CODE HERE
  %%% 找到第一步的动作对应的最大值和索引
  [maxq, a] = max(q(s, :));
  %%% 如果相同则随机
  if q(s, 1) == q(s, 2)
      a = ceil(rand * num_actions);
  end
  %%% 更新步数
  steps = 0;

  %%% 如果未吸收
  while(~absorb)
      %%% 找到下一步的位置
      [x, sn, absorb] =  mountain_car(x, actions(a));
      %%% 奖励
      reward = - double(~ absorb);
      %%% 找到动作对应的最大值和索引
      [maxq, an] = max(q(sn, :));
      %%% 如果相同则随机
      if q(s, 1) == q(s, 2)
        an = ceil(rand * num_actions);
      end
      %%% 找到最大的行动
      q(s, a) = (1 - alpha) * q(s, a) + alpha * (reward + gamma * maxq);
      %%% 更新状态
      a = an;
      s = sn;
      steps = steps + 1;
  end
  %%% 记录步数
  steps_per_episode(i) = steps;
end
```