EM and GMM

> Pattern Recognition Homeworks
>
> Student: thu-zxs

Solutions

EM and Gradient Descent

First check the part of EM step for μ_k, where:

$$\begin{aligned}\tilde{z}_{ik}^{(t+0.5)} &= Prob(x_i \in cluster_k | \{(\mu_j^{(t)}, (\sigma_j^2)^{(t)})\}_{j=1}^K, x_i) \\ &= \frac{\pi_k N(x_i|\mu_k^{(t)}, (\sigma_k^2)^{(t)} I)}{\sum_{j=1}^K \pi_j N(x_i|\mu_j^{(t)}, (\sigma_j^2)^{(t)} I)}\end{aligned}$$

and thus the expectation to be maximize is:

$$Q = \sum_i^n \sum_k^K \frac{\pi_k N(x_i|\mu_k^{(t)}, (\sigma_k^2)^{(t)} I)}{\sum_{j=1}^K \pi_j N(x_i|\mu_j^{(t)}, (\sigma_j^2)^{(t)} I)} (\log N(x_i|\mu_k, (\sigma_k^2)^{(t)} I) + \log \pi_k)$$

noticing that each term of k is independent, so take derivatives of the term w.r.t μ_k, resulting in:

$$\nabla_{\mu_k} Q = \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} * \frac{(\mu_k - x_i)}{-(\sigma_k^2)^{(t)}}$$

taking $\nabla_{\mu_k} Q = 0$ we get:

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n x_i \tilde{z}_{ik}^{(t+0.5)}}{\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)}}$$

Then by Gradient Descent method:

$$\nabla_{\mu_k} l(\{\mu_k^{(t)}, (\sigma_k^2)^{(t)}\}_{k=1}^K) = \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} * \frac{(\mu_k^{(t)} - x_i)}{-(\sigma_k^2)^{(t)}}$$

Take

$$\eta_k^{(t)} = \frac{(\sigma_k^2)^{(t)}}{\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)}}$$

then

$$
\begin{aligned}
\mu_k^{(t+1)} &= \mu_k^{(t)} + \eta_k^{(t)} \nabla_{\mu_k} l(\{\mu_k^{(t)}, (\sigma_k^2)^{(t)}\}_{k=1}^K) \\
&= \mu_k^{(t)} + \frac{(\sigma_k^2)^{(t)}}{\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)}} (\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \frac{\mu_k^{(t)}}{-(\sigma_k^2)^{(t)}} + \frac{\sum_{i=1}^n x_i \tilde{z}_{ik}^{(t+0.5)}}{(\sigma_k^2)^{(t)}}) \\
&= \frac{\sum_{i=1}^n x_i \tilde{z}_{ik}^{(t+0.5)}}{\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)}}
\end{aligned}
$$

which yield the same result as equation (5) by EM.

For σ_k:

$$
\begin{aligned}
Q &= \sum_i^n \sum_k^K \tilde{z}_{ik}^{(t+1)} (\log N(x_i | \mu_k^{(t+1)}, \sigma_k^2 I) + \log \pi_k) \\
\nabla_{\sigma_k^2} Q &= \sum_i^n \tilde{z}_{ik}^{(t+1)} (\frac{(x_i - \mu_k^{(t+1)})^T (x_i - \mu_k^{(t+1)})}{2\sigma_k^4} - \frac{D}{2} \frac{1}{\sigma_k^2})
\end{aligned}
$$

taking $\nabla_{sigma_k^2} Q = 0$ we get:

$$
(\sigma_k^2)^{(t+1)} = \frac{\sum_{i=1}^n \tilde{z}_{ik}^{(t+1)} (x_i - \mu)^T (x_i - \mu)}{D \sum_{i=1}^n \tilde{z}_{ik}^{(t+1)}}
$$

Then by Gradient Descent method:

$$
\nabla_{\sigma_k^2} l(\{\mu_k^{(t+1)}, (\sigma_k^2)^{(t)}\}_{k=1}^K) = \sum_{i=1}^n \tilde{z}_{ik}^{(t+1)} [-\frac{D}{2(\sigma_k^2)^{(t)}} + \frac{(x_i - \mu_k^{(t+1)})^T (x_i - \mu_k^{(t+1)})}{2(\sigma_k^4)^{(t)}}]
$$

take

$$
s_k^{(t)} = \frac{2(\sigma_k^4)^{(t)}}{D \sum_{i=1}^n \tilde{z}_{ik}^{(t+1)}}
$$

Then

$$
\begin{aligned}
(\sigma_k^2)^{(t+1)} &= (\sigma_k^2)^{(t)} + s_k^{(t)} \nabla_{\sigma_k^2} l(\{\mu_k^{(t+1)}, (\sigma_k^2)^{(t)}\}_{k=1}^K) \\
&= \frac{\sum_{i=1}^n \tilde{z}_{ik}^{(t+1)} (x_i - \mu)^T (x_i - \mu)}{D \sum_{i=1}^n \tilde{z}_{ik}^{(t+1)}}
\end{aligned}
$$

which yield the same result as equation (13) by EM.

EM for MAP Estimation

E-step

Compute Expectation:

$$
\begin{aligned}
Q(\Theta, \Theta^{(i-1)}) &= \mathbb{E}[\log(p(\mathbf{x}, \mathbf{z}|\Theta)p(\Theta))|\mathbf{x}, \Theta^{(i-1)}] \\
&= \int_{\mathbf{z}\in\mathbf{Z}} \log(p(\mathbf{x}, \mathbf{z}|\Theta)p(\Theta)) f(\mathbf{z}|\mathbf{x}, \Theta^{(i-1)}) d\mathbf{z} \\
&= \log(p(\Theta)) + \int_{\mathbf{z}\in\mathbf{Z}} \log(p(\mathbf{x}, \mathbf{z}|\Theta)) f(\mathbf{z}|\mathbf{x}, \Theta^{(i-1)}) d\mathbf{z}
\end{aligned}
$$

M-step

$$
\Theta^{(i)} = \mathrm{argmax}_{\Theta} Q(\Theta, \Theta^{(i-1)})
$$

Programming

Category $w1$

The Likelihood function:

$$
L(\Theta|x) = p(x|\Theta) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)
$$

Assume a Gaussian distribution $\mathbf{x}$ has the form:

$$
\begin{aligned}
\mathbf{x} &= \begin{pmatrix} \mathbf{x_a} \\ \mathbf{x_b} \end{pmatrix} \\
\mu &= \begin{pmatrix} \mu_{\mathbf{a}} \\ \mu_{\mathbf{b}} \end{pmatrix} \\
\mathbf{\Sigma} &= \begin{pmatrix} \mathbf{\Sigma_{aa}} & \mathbf{\Sigma_{ab}} \\ \mathbf{\Sigma_{ba}} & \mathbf{\Sigma_{bb}} \end{pmatrix}
\end{aligned}
$$

From the conditional Gaussian distribution formula, the distribution of $\mathbf{x_a}$ is also a Gaussian distribution given $\mathbf{x_b}$ with mean and variance:

$$
\begin{aligned}
\mu_{a|b} &= \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{x_b} - \mu_b) \\
\Sigma_{a|b} &= \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}
\end{aligned}
$$

So the distribution of hidden variables y_k given x_k and $\Theta^{(i-1)}$: $p(y_k|x_k, \Theta^{(i-1)})$ can be computed with same fomular for $k = 2, 4, 6, 8, 10$ which is the missing x_3 value.

So

$$
\begin{aligned}
Q(\Theta, \Theta^{(i-1)}) &= \int_{\mathbf{y}} \log(p(\mathbf{x}, \mathbf{y}|\Theta)) p(\mathbf{y}|\mathbf{x}, \Theta^{(i-1)}) d\mathbf{y} \\
&= (\int_{\mathbf{y}} (\sum_{j=2,4,6,8,10} \log(p(\mathbf{x}_j, \mathbf{y}_j|\Theta))) \prod_{k=2,4,6,8,10} p(\mathbf{y}_k|\mathbf{x}_k, \Theta^{(i-1)}) d\mathbf{y}) + (\sum_{j=1,3,5,7,9} \log(p(\mathbf{x}_j|\Theta))) \\
&= \sum_{j=2,4,6,8,10} \int_{\mathbf{y}_j} (\log(p(\mathbf{x_j}, \mathbf{y_j}|\Theta)) p(\mathbf{y}_j|\mathbf{x}_j, \Theta^{(i-1)}) d\mathbf{y}_j + (\sum_{j=1,3,5,7,9} \log(p(\mathbf{x}_j|\Theta))) \\
&= \sum_{j=2,4,6,8,10} \log(\frac{1}{(2\pi)^{3/2}|\Sigma|^{1/2}}) \int_{\mathbf{y_j}} p(\mathbf{y}_j|\mathbf{x}_j, \Theta^{(i-1)}) d\mathbf{y}_j \\
&\qquad - \frac{1}{2} \int_{\mathbf{y}_j} (\mathbf{x}_j - \mu)^T \Sigma^{-1} (\mathbf{x}_j - \mu) p(\mathbf{y}_j|\mathbf{x}_j, \Theta^{(i-1)}) d\mathbf{y}_j \\
&\qquad + (\sum_{j=1,3,5,7,9} \log(p(\mathbf{x}_j|\Theta))) \\
&= \sum_{j=2,4,6,8,10} \log(\frac{1}{(2\pi)^{3/2}|\Sigma|^{1/2}}) - \frac{1}{2}(\hat{\mathbf{x}}_j - \mu)^T \Sigma^{-1} (\hat{\mathbf{x}}_j - \mu) - \frac{1}{2}\Lambda_{33} (\sigma^2_{a|b})^{(i-1)}_{(j)} \\
&\qquad\qquad + (\sum_{j=1,3,5,7,9} \log(p(\mathbf{x}_j|\Theta))) \\
&= 10 \log(\frac{1}{(2\pi)^{3/2}|\Sigma|^{1/2}}) - \frac{1}{2} \sum_{j=2,4,6,8,10} (\hat{\mathbf{x}}_j - \mu)^T \Sigma^{-1} (\hat{\mathbf{x}}_j - \mu) - \frac{1}{2}\Lambda_{33} \sum_{j=2,4,6,8,10} (\sigma^2_{a|b})^{(i-1)}_{(j)} \\
&\qquad - \frac{1}{2} \sum_{j=1,3,5,7,9} (\mathbf{x}_j - \mu)^T \Sigma^{-1} (\mathbf{x}_j - \mu)
\end{aligned}
$$

Where $\Lambda_{33} = (\Sigma^{-1})_{33}$ and

$$
\hat{\mathbf{x}}_j = \begin{pmatrix} x_1^{(j)} \\ x_2^{(j)} \\ (\mu_{a|b})^{(i-1)}_{(j)} \end{pmatrix}
$$

Let

$$
\begin{aligned}
\frac{\partial Q}{\partial \mu} &= \sum_{j=2,4,6,8,10} \Sigma^{-1} (\hat{\mathbf{x}}_j - \mu) + \sum_{j=1,3,5,7,9} \Sigma^{-1} (\mathbf{x}_j - \mu) \\
&= 0
\end{aligned}
$$

then

$$
\mu^{(i)} = \frac{1}{10} \sum_{j=2,4,6,8,10} \hat{\mathbf{x}}_j + \frac{1}{10} \sum_{j=1,3,5,7,9} \mathbf{x}_j
$$

To find $\Sigma^{(i)}$, rewrite Q as:

$$
\begin{aligned}
Q &= 10(\log(\frac{1}{(2\pi)^{3/2}}) + \frac{1}{2}\log(|\Sigma^{-1}|) - \frac{1}{2}\sum_{j=2,4,6,8,10}\mathrm{tr}(\Sigma^{-1}(\hat{\mathbf{x}}_j-\mu)(\hat{\mathbf{x}}_j-\mu)^T) \\
&\quad -\frac{1}{2}\Lambda_{33}\sum_{j=2,4,6,8,10}(\sigma^2_{a|b})^{(i-1)}_{(j)} - \frac{1}{2}\sum_{j=1,3,5,7,9}\mathrm{tr}(\Sigma^{-1}(\mathbf{x}_j-\mu)(\mathbf{x}_j-\mu)^T) \\
&= 10(\log(\frac{1}{(2\pi)^{3/2}}) + \frac{1}{2}\log(|\Sigma^{-1}|) - \frac{1}{2}\sum_{j=2,4,6,8,10}\mathrm{tr}(\Sigma^{-1}\hat{N}_j) \\
&\quad -\frac{1}{2}\Lambda_{33}\sum_{j=2,4,6,8,10}(\sigma^2_{a|b})^{(i-1)}_{(j)} - \frac{1}{2}\sum_{j=1,3,5,7,9}\mathrm{tr}(\Sigma^{-1}N_j)
\end{aligned}
$$

and let

$$
\begin{aligned}
\frac{\partial Q}{\partial \Sigma^{-1}} &= 10\Sigma - 5\mathrm{diag}(\Sigma) - \frac{1}{2}\sum_{j=2,4,6,8,10}(2\hat{N}_j - \mathrm{diag}(\hat{N}_j)) - \frac{1}{2}\sum_{j=1,3,5,7,9}(2N_j - \mathrm{diag}(N_j)) \\
&\quad -\frac{1}{2}\sum_{j=2,4,6,8,10}(\sigma^2_{a|b})^{(i-1)}_{(j)}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 0
\end{aligned}
$$

then

$$
\begin{aligned}
&\Sigma^{(i)} - \frac{1}{2}\mathrm{diag}(\Sigma^{(i)}) \\
&= \frac{1}{20}(\sum_{j=2,4,6,8,10}(2\hat{N}_j^{(i)} - \mathrm{diag}(\hat{N}_j^{(i)})) + \sum_{j=1,3,5,7,9}(2N_j^{(i)} - \mathrm{diag}(N_j^{(i)})) \\
&+ \sum_{j=2,4,6,8,10}(\sigma^2_{a|b})^{(i-1)}_{(j)})\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix})
\end{aligned}
$$

So

$$
\Sigma^{(i)} = \frac{1}{10}\sum_{j=2,4,6,8,10}(\hat{N}_j^{(i)} + (\sigma^2_{a|b})^{(i-1)}_{(j)}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}) + \sum_{j=1,3,5,7,9}N_j^{(i)}
$$

By above update formular deduction result, **EM** algorithm can be implemented as follows:

1) Initialization

```
import numpy as np
from numpy.linalg import inv

X = np.array([
    [ 0.42, -0.2, 1.3, 0.39, -1.6, -0.029, -0.23, 0.27, -1.9, 0.87 ],
    [-0.087, -3.3, -0.32, 0.71, -5.3, 0.89, 1.9, -0.3, 0.076, -1.0 ],
    [0.58, -3.4, 1.7, 0.23, -0.15, -4.7, 2.2, -0.87, -2.1, -2.6 ] ])
mu = np.zeros((3,1))
Sigma = np.eye(3)
```

```
epsilon = 1e-10
mu_c = np.zeros((5, 1))
sigma_c = np.zeros((5, 1))
x_hat = np.zeros((3, 5))
N_hat = np.zeros((5, 3, 3))
N = np.zeros((5, 3, 3))
```

2) **EM** for missing data

```
error = 1
while error > epsilon:
    for j in xrange(1, 11, 2):
        mu_c[j/2, 0] = mu[2,0] + np.squeeze(Sigma[2:3, 0:2].dot(inv(Sigma[0:2,
0:2])).dot(X[0:2, j:j+1]-mu[0:2, 0:1]))
        sigma_c[j/2, 0] = Sigma[2,2] - np.squeeze(Sigma[2:3,
0:2].dot(inv(Sigma[0:2,0:2])).dot(Sigma[0:2,2:3]))
        x_hat[:, j/2] = np.array([X[0, j],X[1, j], mu_c[j/2,0]])

    mu_new = 1.0/10 * (np.sum(x_hat, axis=1) + np.sum(X[:, ::2], axis=1))[:,
np.newaxis]
    error = np.sum(np.abs(mu_new - mu))
    mu = mu_new

    for j in xrange(1, 11, 2):
        N_hat[j/2, :, :] = (x_hat[:, j/2]-mu).dot((x_hat[:, j/2]-mu).T)
        N[j/2, :, :] = (X[:, j-1]-mu).dot((X[:, j-1]-mu).T)

    Sigma_tmp = 1.0/10 * (np.sum(N_hat, axis=0) + np.sum(N, axis=0))
    Sigma_tmp[2,2] += 1.0/10*np.sum(sigma_c)
    error = np.sum(np.abs(Sigma_tmp-Sigma))
    Sigma = Sigma_tmp

print("data missing (use EM Algorithm): \n -------------")
print("mu:")
print(mu)
print("Sigma:")
print(Sigma)
print("\n")
```

Which yield the results below after 48 iterations:

```
data missing (use EM Algorithm):
 -------------
mu:
[[-0.0709     ]
 [-0.6731     ]
 [ 1.41273029]]
Sigma:
[[ 16.13360535  16.66440267  14.82588868]
 [ 16.66440267  18.28313451  12.67635952]
 [ 14.82588868  12.67635952  20.12164849]]
```

When there is no missing data, by **MLE**:

```
mu = np.mean(X, axis=1)[:, np.newaxis]
Sigma = 1.0/10 * ((X-mu).dot((X-mu).T))
```

yield the result of :

```
No missing data (use max-likelihood method):
 --------------
mu:
[[-0.0709]
 [-0.6731]
 [-0.911 ]]
Sigma:
[[ 0.90617729  0.69289221  0.3940801 ]
 [ 0.69289221  4.05613089  0.8150299 ]
 [ 0.3940801   0.8150299   4.541949  ]]
```

Category $w2$

$$
\begin{aligned}
Q(\Theta, \Theta^{(i-1)}) &= \int_{\mathbf{y}} \log(p(\mathbf{x}, \mathbf{y}|\Theta))p(\mathbf{y}|\mathbf{x}, \Theta^{(i-1)})d\mathbf{y} \\
&= (\int_{\mathbf{y}} (\sum_{j=2,4,6,8,10} \log(p(\mathbf{x}_j, \mathbf{y}_j|\Theta))) \prod_{k=2,4,6,8,10} p(\mathbf{y}_k|\mathbf{x}_k, \Theta^{(i-1)})d\mathbf{y}) + (\sum_{j=1,3,5,7,9} \log(p(\mathbf{x}_j|\Theta))) \\
&= \sum_{j=2,4,6,8,10} \int_{\mathbf{y}_j} (\log(p(\mathbf{x_j}, \mathbf{y_j}|\Theta))p(\mathbf{y}_j|\mathbf{x}_j, \Theta^{(i-1)})d\mathbf{y}_j + (\sum_{j=1,3,5,7,9} \log(p(\mathbf{x}_j|\Theta)))
\end{aligned}
$$

To maximize this formula, firstly choose $\mathbf{y}_j$ to be inside the range parameterized by $\Theta^{(i-1)}$, then choose Θ to cover $\max(\mathbf{x}_j)$ and $\min(\mathbf{x}_j)$.

This is a one-step algorithm. Let's say, choosing $\mathbf{y}_j$ to be $(x_l)_{(3)}^{(i-1)}$ or $(x_r)_{(3)}^{(i-1)}$, i.e. $\mathbf{y}_j \in \{(x_l)_{(3)}^{(i-1)}, (x_r)_{(3)}^{(i-1)}\}$:

1) Initialization

```
X = np.array([
        [-0.4, -0.31, 0.38,-0.15, -0.35, 0.17, -0.011, -0.27, -0.065, -0.12],
        [0.58, 0.27, 0.055, 0.53, 0.47, 0.69, 0.55, 0.61, 0.49, 0.054],
        [0.089, -0.04, -0.035, 0.011, 0.034, 0.1, -0.18, 0.12, 0.0012, -0.063]
])

xl = np.array([[-2], [-2], [-2]], dtype=np.float32)
xr = np.array([[2], [2], [2]], dtype=np.float32)
```

2) **EM** for missing data case

```
xl[0, 0] = np.min(X[0, :]); xr[0, 0] = np.max(X[0, :])
xl[1, 0] = np.min(X[1, :]); xr[1, 0] = np.max(X[1, :])

print("data missing \n --------")
print("xl={}".format(xl))
print("xr={}".format(xr))
```

3) No missing data

```
print("no missing data \n --------")
print("xl={}".format(np.min(X, axis=1)[:, np.newaxis]))
print("xr={}".format(np.max(X, axis=1)[:, np.newaxis]))
```

result:

```
===== Uniform Distribution =====
data missing
 --------
xl=[[-0.40000001]
 [ 0.054     ]
 [-2.        ]]
xr=[[ 0.38]
 [ 0.69]
 [ 2.  ]]
no missing data
 --------
xl=[[-0.4  ]
 [ 0.054]
 [-0.18 ]]
xr=[[ 0.38]
 [ 0.69]
 [ 0.12]]
```