Expectation-Maximization (EM)

Expectation–Maximization (EM) algorithm. Let X be a random variable distributed according to $p_X(x)$ and Y be a random variable distributed according to $p_Y(y)$. Let $\mathrm{D}_X = \{x_i\}_{i=1}^m$ be an i.i.d~sample from $p_X(x)$ and $\mathrm{D}_Y = \{y_i\}_{i=1}^n$ be an i.i.d.~sample from $p_Y(y)$. Finally, let $p_X(x)$ and $p_Y(y)$ be defined as follows

$$p_X(x) = \alpha \mathrm{N}(\mu_1, \sigma_1^2) + (1-\alpha)\mathrm{N}(\mu_2, \sigma_2^2)$$

and

$$p_Y(y) = \beta \mathrm{N}(\mu_1, \sigma_1^2) + (1-\beta)\mathrm{N}(\mu_2, \sigma_2^2),$$

where $\mathrm{N}(\mu, \sigma^2)$ is a univariate Gaussian distribution with mean μ and variance σ^2, $\alpha \in (0,1)$, $\beta \in (0,1)$, $\mu_1 \in \mathbb{R}$, $\mu_2 \in \mathbb{R}$, $\sigma_1 \in \mathbb{R}^+$ and $\sigma_2 \in \mathbb{R}^+$ are unknown parameters.

a) Derive update rules of the EM algorithm for estimating $\beta, \mu_1, \mu_2, \sigma_1$, and σ_2 based only on data set D_Y.

Ans:

To solve this problem in more general way,
now set

1. Y as observed variable: $Y|Z = C_k \sim \mathrm{N}(Y|\mu_k, \sigma_k^2)$
2. Z as latent variable: $\sum_{k=1}^K p_k = 1$

Z	1	2	3	4	... K
P	p_1	p_2	p_3	p_4	$\cdots$ p_K

3. θ as parameters: $\theta = \{p_1 \dots p_k, \mu_1 \dots \mu_k, \sigma_1 \dots \sigma_k\}$

A. E–step:

$Q(\theta, \theta^{(t)}) = E[logP(Y, Z|\theta)|Y, \theta^{(t)}]$

Before calculating Q, first find several distributions:

$$P(Y|\theta) = \sum_{k=1}^K p_k \mathrm{N}(Y|\mu_k, \sigma_k^2)$$

$$\begin{aligned} P(Y, Z|\theta) &= P(Z|\theta)P(Y|Z, \theta) \\ &= p_Z \mathrm{N}(Y|\mu_Z, \sigma_Z^2) \end{aligned}$$

$$
\begin{aligned}
P(Z|Y,\theta^{(t)}) &= \frac{P(Y,Z|\theta^{(t)})}{P(Y|\theta^{(t)})} \\
&= \frac{p_Z^{(t)}\mathrm{N}(Y|\mu_Z^{(t)},\sigma_Z^{2(t)})}{\sum_{k=1}^{K} p_k^{(t)}\mathrm{N}(Y|\mu_k^{(t)},\sigma_k^{2(t)})}
\end{aligned}
$$

Now the expectation is:

$$
\begin{aligned}
Q(\theta,\theta^{(t)}) &= E[logP(Y,Z|\theta)|Y,\theta^{(t)}] \\
&= \sum_{Z} \log \prod_{i=1}^{n} p(y_i,z_i|\theta) \prod_{i=1}^{n} p(z_i|y_i,\theta^{(t)}) \\
&= \sum_{Z} [\sum_{i=1}^{n} \log p(y_i,z_i|\theta)] \prod_{i=1}^{n} p(z_i|y_i,\theta^{(t)}) \\
&= \sum_{i=1}^{n} \sum_{z_i} \log(p(y_i,z_i|\theta)) p(z_i|y_i,\theta^{(t)}) \\
&= \sum_{i=1}^{n} \sum_{z_i} \log(p_{z_i}\mathrm{N}(y_i|\mu_{z_i},\sigma_{z_i}^2)) p(z_i|y_i,\theta^{(t)}) \\
&= \sum_{k=1}^{K} \sum_{i=1}^{n} [\log(p_k) p(k|y_i,\theta^{(t)}) + \log(\mathrm{N}(y_i|\mu_k,\sigma_k^2))] p(k|y_i,\theta^{(t)})
\end{aligned}
$$

where

$$
\begin{aligned}
p(z_i|y_i,\theta^{(t)}) &= \frac{p_{z_i}^{(t)}\mathrm{N}(y_i|\mu_{z_i}^{(t)},\sigma_{z_i}^{2(t)})}{\sum_{k=1}^{K} p_k^{(t)}\mathrm{N}(y_i|\mu_k^{(t)},\sigma_k^{2(t)})} \\
p(z_i=k|y_i,\theta^{(t)}) &= \frac{p_k^{(t)}\mathrm{N}(y_i|\mu_k^{(t)},\sigma_k^{2(t)})}{\sum_{k=1}^{K} p_k^{(t)}\mathrm{N}(y_i|\mu_k^{(t)},\sigma_k^{2(t)})}
\end{aligned}
$$

B. M–step

1. To maximize $Q(\theta,\theta^{(t)})$ with respect to p, we observe that this is an instance of constrained optimization because it must hold that $\sum_{k=1}^{K} p_k = 1$ and $p_k \geq 0$. We will use the method of Lagrange multipliers:

$$
L(p,\lambda) = \sum_{k=1}^{K} \sum_{i=1}^{n} \log(p_k) p(k|y_i,\theta^{(t)}) + \lambda(\sum_{k=1}^{K} p_k - 1)
$$

$$
\begin{aligned}
\frac{\partial L(p,\lambda)}{\partial p_k} &= \frac{\sum_{i=1}^{n} p(k|y_i,\theta^{(t)})}{p_k} + \lambda = 0 \\
\sum_{i=1}^{n} p(k|y_i,\theta^{(t)}) + p_k\lambda &= 0 \\
\sum_{i=1}^{n} \underbrace{\sum_{k=1}^{K} p(k|y_i,\theta^{(t)})}_{1} + \underbrace{\sum_{k=1}^{K} p_k}_{1} \lambda &= 0 \\
\sum_{i=1}^{n} 1 + \lambda &= 0 \\
\lambda &= -n
\end{aligned}
$$

So that

$$p_k^{(t+1)} = \frac{\sum_{i=1}^{n} p(k|y_i, \theta^{(t)})}{n}$$

In this question, $p_1 = \beta$, so

$$\beta^{(t+1)} = \frac{\sum_{i=1}^{n} p(z_i = 1|y_i, \theta^{(t)})}{n}$$

2. To maximize $Q(\theta, \theta^{(t)})$ with respect to μ:

$$\begin{aligned} L(\mu) &= \sum_{k=1}^{K}\sum_{i=1}^{n}[\log(\mathrm{N}(y_i|\mu_k, \sigma_k^2))p(k|y_i, \theta^{(t)})] \\ &= \sum_{k=1}^{K}\sum_{i=1}^{n} -[\frac{(y_i - \mu_k)^2}{2\sigma_k^2}p(k|y_i, \theta^{(t)})] \end{aligned}$$

$$\frac{\partial L(\mu)}{\partial \mu_k} = \frac{1}{\sigma_k^2}\sum_{i=1}^{n}[(y_i - \mu_k)p(k|y_i, \theta^{(t)})] = 0$$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{n} y_i p(k|y_i, \theta^{(t)})}{\sum_{i=1}^{n} p(k|y_i, \theta^{(t)})}$$

In this question:

$$\begin{aligned} \mu_1^{(t+1)} &= \frac{\sum_{i=1}^{n} y_i p(z_i = 1|y_i, \theta^{(t)})}{\sum_{i=1}^{n} p(z_i = 1|y_i, \theta^{(t)})} \\ \mu_2^{(t+1)} &= \frac{\sum_{i=1}^{n} y_i p(z_i = 2|y_i, \theta^{(t)})}{\sum_{i=1}^{n} p(z_i = 2|y_i, \theta^{(t)})} \end{aligned}$$

3. To maximize $Q(\theta, \theta^{(t)})$ with respect to σ:

$$\begin{aligned} L(\sigma) &= \sum_{k=1}^{K}\sum_{i=1}^{n}[\log(\mathrm{N}(y_i|\mu_k, \sigma_k^2))p(k|y_i, \theta^{(t)})] \\ &= \sum_{k=1}^{K}\sum_{i=1}^{n} -[\frac{(y_i - \mu_k)^2}{2\sigma_k^2}p(k|y_i, \theta^{(t)}) + \frac{1}{2}\log(2\pi\sigma_k^2)p(k|y_i, \theta^{(t)})] \end{aligned}$$

$$\frac{\partial L(\sigma)}{\partial \sigma_k^2} = \sum_{i=1}^{n}\{[(y_i - \mu_k)^2 {\sigma_k^2}^{-2} - {\sigma_k^2}^{-1}]p(k|y_i, \theta^{(t)})\} = 0$$

$$\sigma_k^{2(t+1)} = \frac{\sum_{i=1}^{n}[(y_i - \mu_k)^2 p(k|y_i, \theta^{(t)})]}{\sum_{i=1}^{n} p(k|y_i, \theta^{(t)})}$$

In this question:

$$\sigma_1^{2(t+1)} = \frac{\sum_{i=1}^{n}[(y_i - \mu_1)^2 p(z_i = 1|y_i, \theta^{(t)})]}{\sum_{i=1}^{n} p(z_i = 1|y_i, \theta^{(t)})}$$
$$\sigma_2^{2(t+1)} = \frac{\sum_{i=1}^{n}[(y_i - \mu_2)^2 p(z_i = 2|y_i, \theta^{(t)})]}{\sum_{i=1}^{n} p(z_i = 2|y_i, \theta^{(t)})}$$

b) Derive update rules of an EM algorithm for estimating $\alpha, \beta, \mu_1, \mu_2, \sigma_1$, and σ_2 based on data sets D_X and D_Y.

Ans:

A. E–step:

$$Q(\theta, \theta^{(t)}) = E[logP(X, Z|\theta)|X, \theta^{(t)}] + E[logP(Y, Z|\theta)|Y, \theta^{(t)}]$$

Now the expectation is:

$$\begin{aligned} Q(\theta, \theta^{(t)}) &= E[logP(X, Z|\theta)|X, \theta^{(t)}] + E[logP(Y, Z|\theta)|Y, \theta^{(t)}] \\ &= \sum_{k=1}^{K}\sum_{i=1}^{m}[\log(\alpha_k)p(k|x_i, \theta^{(t)}) + \log(\mathrm{N}(x_i|\mu_k, \sigma_k^2))p(k|x_i, \theta^{(t)})] + \\ &\quad \sum_{k=1}^{K}\sum_{j=1}^{n}[\log(\beta_k)p(k|y_j, \theta^{(t)}) + \log(\mathrm{N}(y_j|\mu_k, \sigma_k^2))p(k|y_j, \theta^{(t)})] \end{aligned}$$

where

$$p(z_i = k|x_i, \theta^{(t)}) = \frac{\alpha_k^{(t)}\mathrm{N}(x_i|\mu_k^{(t)}, \sigma_k^{2(t)})}{\sum_{k=1}^{K}\alpha_k^{(t)}\mathrm{N}(x_i|\mu_k^{(t)}, \sigma_k^{2(t)})}$$
$$p(z_i = k|y_j, \theta^{(t)}) = \frac{\beta_k^{(t)}\mathrm{N}(y_j|\mu_k^{(t)}, \sigma_k^{2(t)})}{\sum_{k=1}^{K}\beta_k^{(t)}\mathrm{N}(y_j|\mu_k^{(t)}, \sigma_k^{2(t)})}$$
$$\sum_{k=1}^{K}\alpha_k = 1$$
$$\sum_{k}^{K}\beta_k = 1$$

B. M–step

1. To maximize $Q(\theta, \theta^{(t)})$ with respect to α and β. We will use the method of Lagrange multipliers:

$$\alpha^{(t+1)} = \frac{\sum_{i=1}^{m} p(z_i = 1|x_i, \theta^{(t)})}{m}$$
$$\beta^{(t+1)} = \frac{\sum_{j=1}^{n} p(z_j = 1|y_j, \theta^{(t)})}{n}$$

2. To maximize $Q(\theta, \theta^{(t)})$ with respect to μ:

$$
\begin{aligned}
L(\mu) &= \sum_{k=1}^{K}\sum_{i=1}^{m}[\log(\mathrm{N}(x_i|\mu_k,\sigma_k^2))p(k|x_i,\theta^{(t)})] + \sum_{k=1}^{K}\sum_{j=1}^{n}[\log(\mathrm{N}(y_j|\mu_k,\sigma_k^2))p(k|y_j,\theta^{(t)})] \\
&= \sum_{k=1}^{K}\sum_{i=1}^{m} -[\frac{(x_i-\mu_k)^2}{2\sigma_k^2}p(k|x_i,\theta^{(t)})] + \sum_{k=1}^{K}\sum_{j=1}^{n} -[\frac{(y_j-\mu_k)^2}{2\sigma_k^2}p(k|y_j,\theta^{(t)})]
\end{aligned}
$$

$$
\frac{\partial L(\mu)}{\partial \mu_k} = \frac{1}{\sigma_k^2}\sum_{i=1}^{m}[(x_i-\mu_k)p(k|x_i,\theta^{(t)})] + \frac{1}{\sigma_k^2}\sum_{j=1}^{n}[(y_j-\mu_k)p(k|y_j,\theta^{(t)})] = 0
$$

$$
\mu_k^{(t+1)} = \frac{\sum_{i=1}^{m} x_i p(k|x_i,\theta^{(t)}) + \sum_{j=1}^{n} y_j p(k|y_j,\theta^{(t)})}{\sum_{i=1}^{m} p(k|x_i,\theta^{(t)}) + \sum_{j=1}^{n} p(k|y_j,\theta^{(t)})}
$$

In this question:

$$
\begin{aligned}
\mu_1^{(t+1)} &= \frac{\sum_{i=1}^{m} x_i p(z_i=1|x_i,\theta^{(t)}) + \sum_{j=1}^{n} y_j p(z_i=1|y_j,\theta^{(t)})}{\sum_{i=1}^{m} p(z_i=1|x_i,\theta^{(t)}) + \sum_{j=1}^{n} p(z_i=1|y_j,\theta^{(t)})} \\
\mu_2^{(t+1)} &= \frac{\sum_{i=1}^{m} x_i p(z_i=2|x_i,\theta^{(t)}) + \sum_{j=1}^{n} y_j p(z_i=2|y_j,\theta^{(t)})}{\sum_{i=1}^{m} p(z_i=2|x_i,\theta^{(t)}) + \sum_{j=1}^{n} p(z_i=2|y_j,\theta^{(t)})}
\end{aligned}
$$

3. To maximize $Q(\theta,\theta^{(t)})$ with respect to σ:

$$
\begin{aligned}
L(\sigma) &= \sum_{k=1}^{K}\sum_{i=1}^{m}[\log(\mathrm{N}(x_i|\mu_k,\sigma_k^2))p(k|x_i,\theta^{(t)})] + \sum_{k=1}^{K}\sum_{j=1}^{n}[\log(\mathrm{N}(y_j|\mu_k,\sigma_k^2))p(k|y_j,\theta^{(t)})] \\
&= \sum_{k=1}^{K}\sum_{i=1}^{m} -[\frac{(x_i-\mu_k)^2}{2\sigma_k^2}p(k|x_i,\theta^{(t)}) + \frac{1}{2}\log(2\pi\sigma_k^2)p(k|x_i,\theta^{(t)})] + \\
&\quad \sum_{k=1}^{K}\sum_{j=1}^{n} -[\frac{(y_j-\mu_k)^2}{2\sigma_k^2}p(k|y_j,\theta^{(t)}) + \frac{1}{2}\log(2\pi\sigma_k^2)p(k|y_j,\theta^{(t)})]
\end{aligned}
$$

$$
\frac{\partial L(\sigma)}{\partial \sigma_k^2} = \sum_{i=1}^{m}\{[(x_i-\mu_k)^2{\sigma_k^2}^{-2} - {\sigma_k^2}^{-1}]p(k|x_i,\theta^{(t)})\} + \sum_{j=1}^{n}\{[(y_j-\mu_k)^2{\sigma_k^2}^{-2} - {\sigma_k^2}^{-1}]p(k|y_j,\theta^{(t)})\} = 0
$$

$$
\sigma_k^{2(t+1)} = \frac{\sum_{i=1}^{n}[(x_i-\mu_k)^2 p(k|x_i,\theta^{(t)})] + \sum_{j=1}^{n}[(y_j-\mu_k)^2 p(k|y_j,\theta^{(t)})]}{\sum_{i=1}^{m} p(k|x_i,\theta^{(t)}) + \sum_{j=1}^{n} p(k|y_j,\theta^{(t)})}
$$

In this question:

$$
\begin{aligned}
\sigma_1^{2(t+1)} &= \frac{\sum_{i=1}^{n}[(x_i-\mu_k)^2 p(z_i=1|x_i,\theta^{(t)})] + \sum_{j=1}^{n}[(y_j-\mu_k)^2 p(z_j=1|y_j,\theta^{(t)})]}{\sum_{i=1}^{m} p(z_i=1|x_i,\theta^{(t)}) + \sum_{j=1}^{n} p(z_j=1|y_j,\theta^{(t)})} \\
\sigma_2^{2(t+1)} &= \frac{\sum_{i=1}^{n}[(x_i-\mu_k)^2 p(z_i=2|x_i,\theta^{(t)})] + \sum_{j=1}^{n}[(y_j-\mu_k)^2 p(z_j=2|y_j,\theta^{(t)})]}{\sum_{i=1}^{m} p(z_i=2|x_i,\theta^{(t)}) + \sum_{j=1}^{n} p(z_j=2|y_j,\theta^{(t)})}
\end{aligned}
$$

c) Implement both learning algorithms from above and evaluate them on simulated data when $m, n = 100$ and $m, n = 1000$. However, in each case repeat the experiment at least $B = 100$ times to estimate the expectation and variance of all parameters. To do so, repeatedly draw samples D_X and D_Y and then estimate the parameters based on these samples. Finally, average those B estimates and calculate their mean and variance. Document all experiments and discuss your findings.

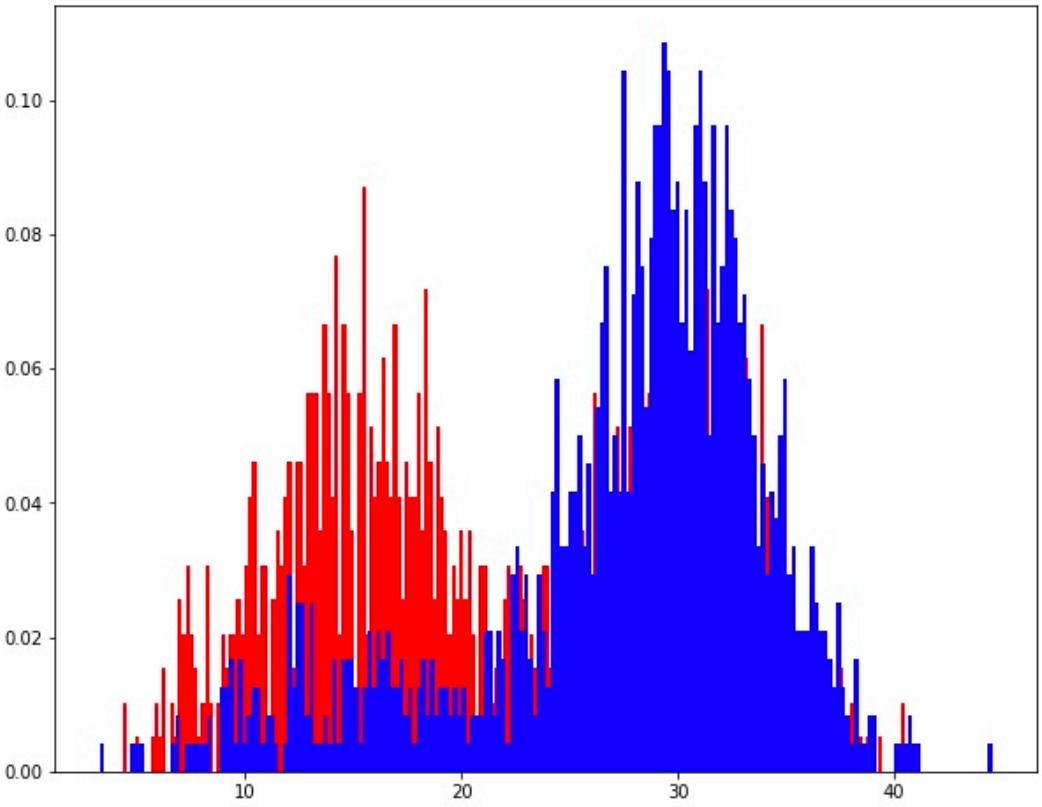
In my experience, the maximum iteration times is 300, and the tolerance level is 10^-9. The stopping criteria are either reaching the max_iter or the sum of absolute difference between time t parameters and time $t-1$ parameters is less than tolerance level.

Experiment 1

$m, n = 1000, 1000$

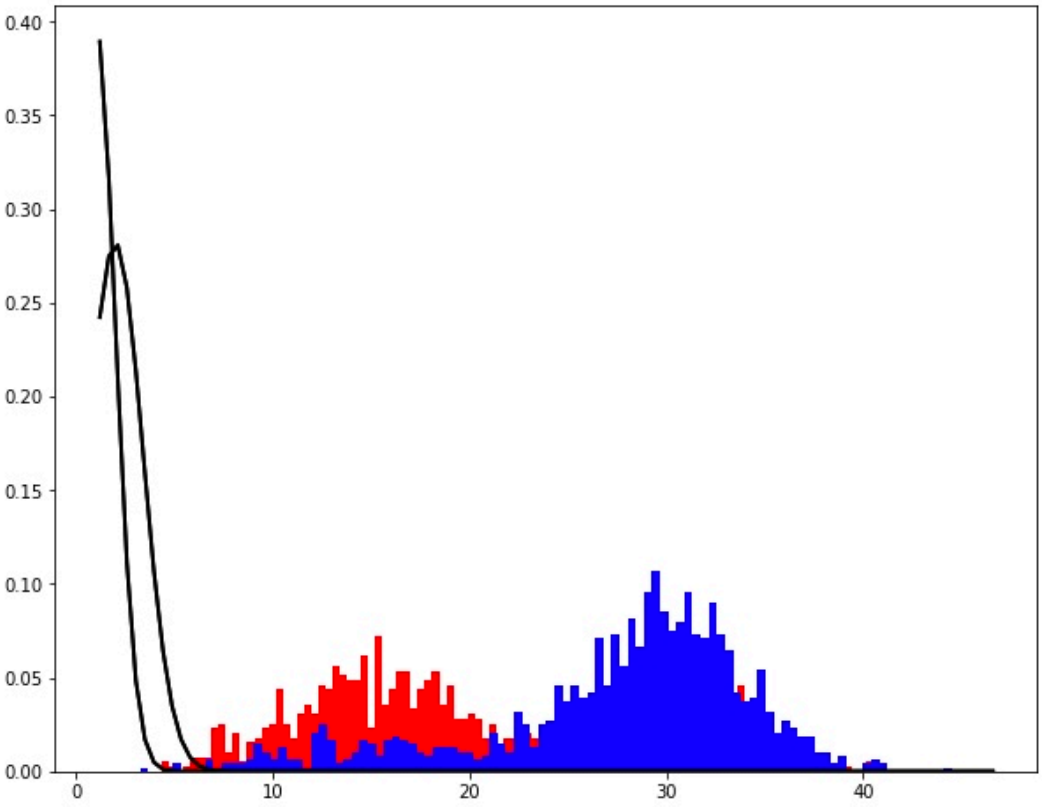
1.Generate Data

$\alpha = 0.6, \beta = 0.2, \mu_1 = 15, \mu_2 = 30, \sigma_1^2 = 20, \sigma_2^2 = 15$



2.Randomly initialize parameters

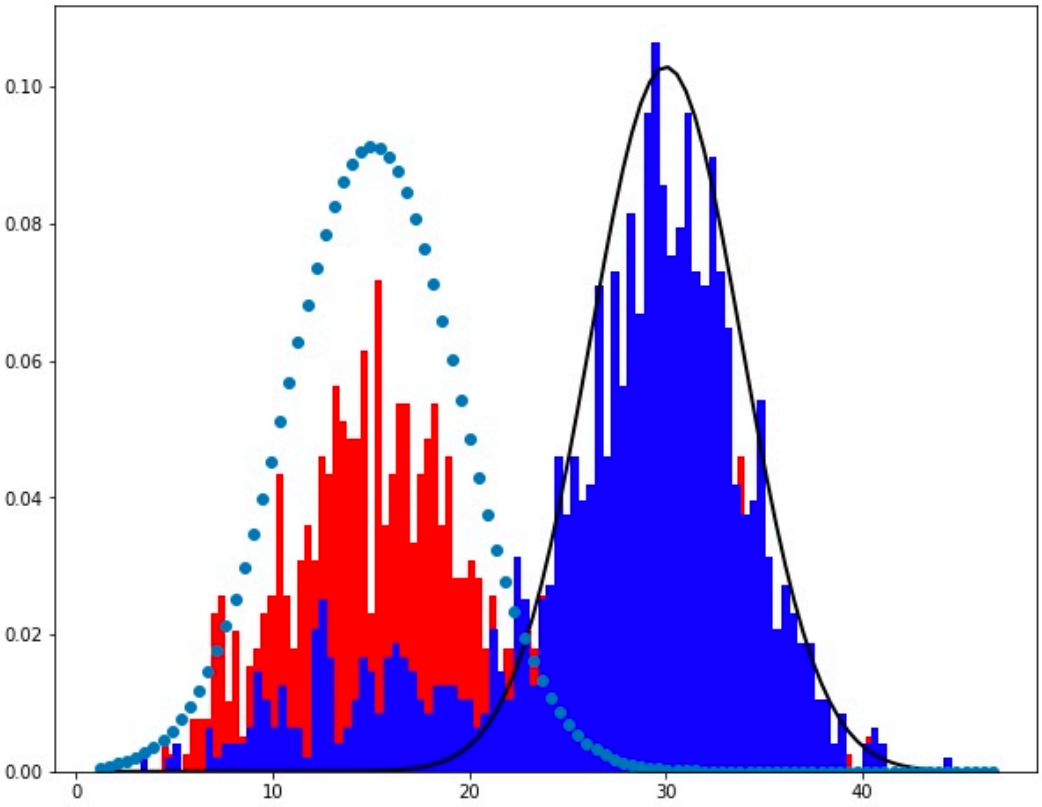
```
# Initialize alpha
Z_x = sorted([random.random(), 1-random.random()])
# Initialize beta
Z_y = sorted([random.random(), 1-random.random()])
# Initialize mu1 and mu2
means = [random.randint(1,30),random.randint(1,20)]
# Initialize sigma_square_1 and sigma_quare_2
S = [random.randint(1,30),random.randint(1,20)]
```



3.Iteratively find the optimal solution

```
for i in range(300):
    # Time t patameters
    Z_x_old = Z_x
    Z_y_old = Z_y
    means_old = means
    S_old = S
    # Updating
    gamma_x = update_gamma(data[0], means, S, Z_x)
    gamma_y = update_gamma(data[1], means, S, Z_y)
    Z_x = update_Z(gamma_x, data[0])
    Z_y = update_Z(gamma_y, data[1])
    means = update_mu(gamma_x, gamma_y, data)
    S = update_sigma(gamma_x, gamma_y, data, means)
    # Difference
    diff = np.nansum(abs(Z_x - Z_x_old)) + np.nansum(abs(Z_y - Z_y_old)) + np.nansum(abs(means - m
eans_old)) + np.nansum(abs(S - S_old))
    # Stop criteria
    if diff <= eps and diff != 0:
        print('Early stopping at step {}, the diff is {}'.format(i+1, diff))
        break
```

Stop criteria are either imposing the maximum number of steps or when the difference of parameter at time t and $t-1$ is less than tolerance level.



4.Repeat the process for 100 times.

5.Evaluation

Evaluate α and β

```
alpha = np.array(alphas)
beta = np.array(betas)

print('For the alpha: the true alpha = {}, mean = {}, variance = {}'.format(true_Z[0][0],np.nanmea
n(alpha), np.nanvar(alpha)))
print('For the beta: the true beta = {}, mean = {}, variance = {}'.format(true_Z[1][0],np.nanmean(
beta), np.nanvar(beta)))
```

```
For the alpha: the true alpha = 0.6, mean = 0.4988789381326482, variance = 0.0031406915499869476
For the beta: the true beta = 0.2, mean = 0.5066957332497739, variance = 0.1120372803102257
```

```
alphas[:10]
```

```
[0.5560530847590975,
 0.44394691505850353,
 0.4439469150990536,
 0.5560530847083869,
 0.5560530847370649,
 0.5560530847396205,
 0.5560530848051715,
 0.4439469150344747,
 0.5560530847367745,
 0.4439469151592899]
```

```
betas[:10]
```

```
[0.16521333176410535,
 0.8347866681141207,
 0.834786668141192,
 0.16521333173025024,
 0.16521333174939626,
 0.1652133317511012,
 0.16521333179486483,
 0.8347866680980788,
 0.16521333174920158,
 0.8347866681814087]
```

Evaluate μ

```
print('The true mean is {}'.format(true_means))
means = np.array(means_s)
#Z_i = 1
print('For the first Gaussian distribution: mean = {}, variance = {}'.format(np.nanmean(means[:,0]
), np.nanvar(means[:,0])))
#Z_i = 2
print('For the second Gaussian distribution: mean = {}, variance = {}'.format(np.nanmean(means[:,1
]), np.nanvar(means[:,1])))
```

```
The true mean is [15 30]
For the first Gaussian distribution: mean = 22.693785276494314, variance = 55.4044335440785
For the second Gaussian distribution: mean = 22.39598909508589, variance = 55.40443347492932
```

```
means_s[:10]
```

```
[array([15.09998261, 29.98979176]),
 array([29.98979176, 15.09998261]),
 array([29.98979176, 15.09998261]),
 array([15.09998261, 29.98979176]),
 array([15.09998261, 29.98979176]),
 array([15.09998261, 29.98979176]),
 array([15.09998261, 29.98979176]),
 array([29.98979176, 15.09998261]),
 array([15.09998261, 29.98979176]),
 array([29.98979176, 15.09998261])]
```

Evalute Variance σ^2

```
print('The true variance is {}'.format(true_S))
S = np.array(Ss)
#Z_i = 1
print('For the first Gaussian distribution: mean = {}, variance = {}'.format(np.nanmean(S[:,0]), n
p.nanvar(S[:,0])))
#Z_i = 2
print('For the second Gaussian distribution: mean = {}, variance = {}'.format(np.nanmean(S[:,1]),
np.nanvar(S[:,1])))
```

```
The true variance is [20 15]
For the first Gaussian distribution: mean = 17.0249963192281, variance = 4.139967455160417
For the second Gaussian distribution: mean = 17.106400270837593, variance = 4.1399675024860265
```

```
Ss[:10]
```

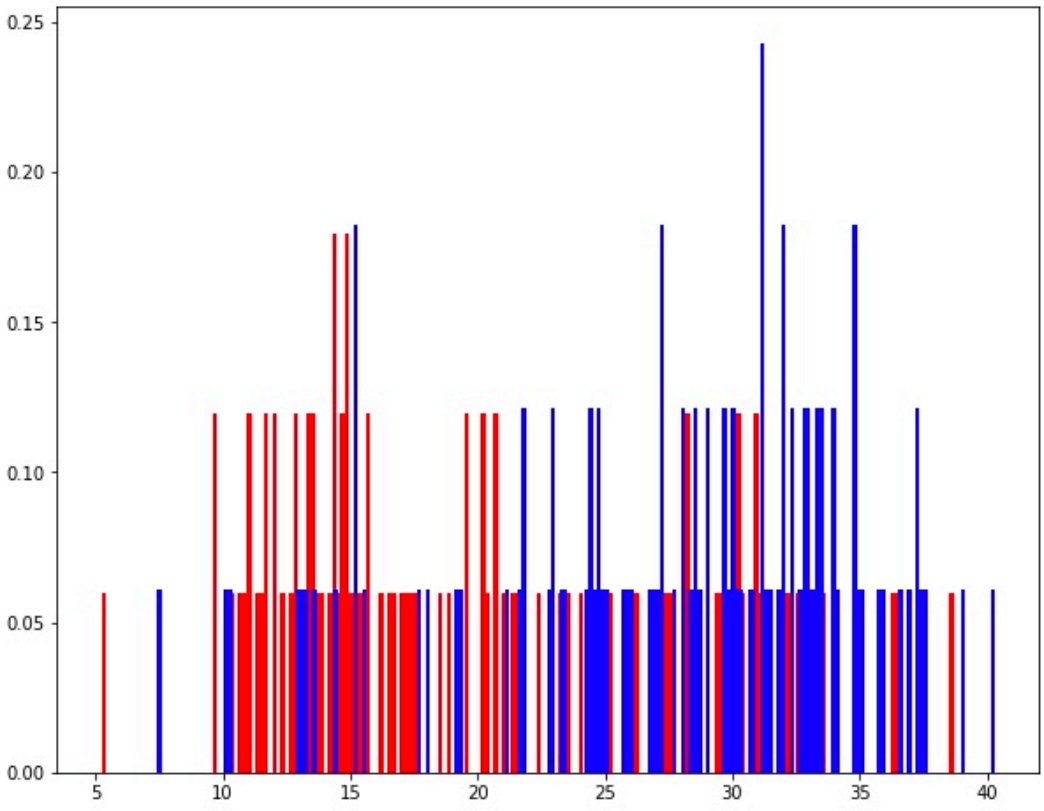
```
[array([19.10079634, 15.03060024]),
 array([15.03060023, 19.10079636]),
 array([15.03060023, 19.10079636]),
 array([19.10079633, 15.03060024]),
 array([19.10079634, 15.03060024]),
 array([19.10079634, 15.03060024]),
 array([19.10079634, 15.03060024]),
 array([15.03060023, 19.10079636]),
 array([19.10079634, 15.03060024]),
 array([15.03060024, 19.10079635])]
```

Experiment 2

$m, n = 100, 100$

1.Generate Data

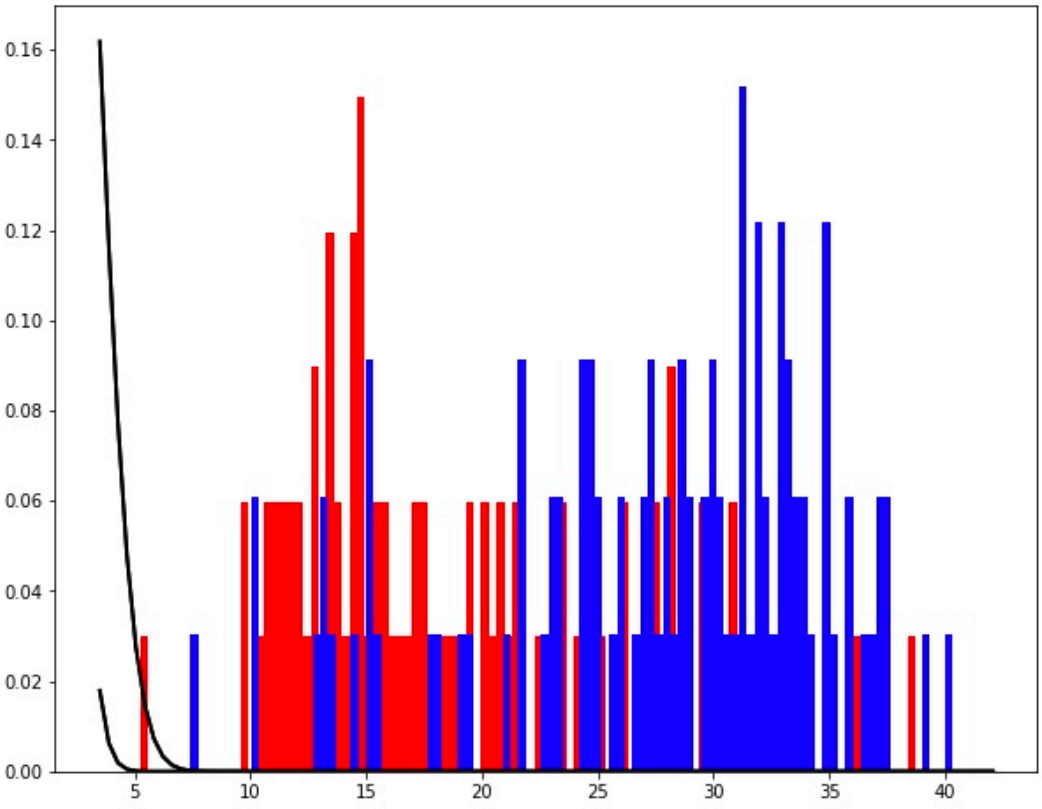
$\alpha = 0.6, \beta = 0.2, \mu_1 = 15, \mu_2 = 30, \sigma_1^2 = 20, \sigma_2^2 = 15$



2.Randomly initialize parameters

```
# Initialize alpha
Z_x = sorted([random.random(), 1-random.random()])
```

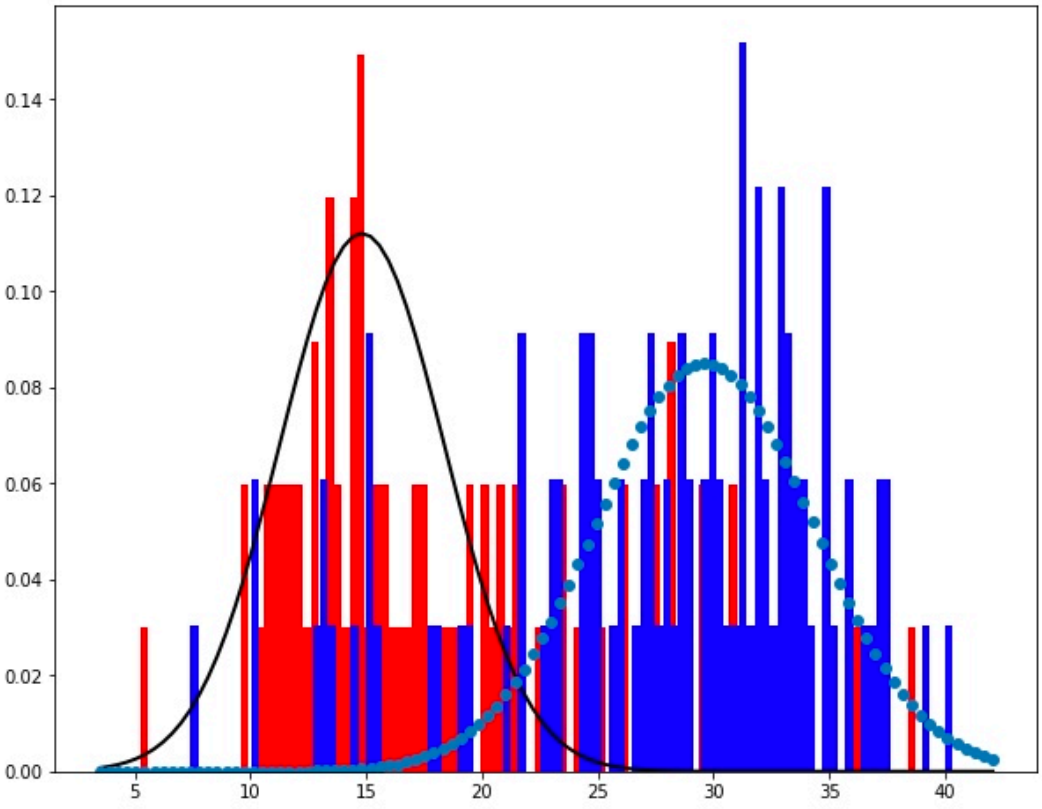
```
# Initialize beta
Z_y = sorted([random.random(), 1-random.random()])
# Initialize mu1 and mu2
means = [random.randint(1,30),random.randint(1,20)]
# Initialize sigma_square_1 and sigma_quare_2
S = [random.randint(1,30),random.randint(1,20)]
```



3.Iteratively find the optimal solution

```
for i in range(300):
    # Time t patameters
    Z_x_old = Z_x
    Z_y_old = Z_y
    means_old = means
    S_old = S
    # Updating
    gamma_x = update_gamma(data[0], means, S, Z_x)
    gamma_y = update_gamma(data[1], means, S, Z_y)
    Z_x = update_Z(gamma_x, data[0])
    Z_y = update_Z(gamma_y, data[1])
    means = update_mu(gamma_x, gamma_y, data)
    S = update_sigma(gamma_x, gamma_y, data, means)
    # Difference
    diff = np.nansum(abs(Z_x - Z_x_old)) + np.nansum(abs(Z_y - Z_y_old)) + np.nansum(abs(means - m
eans_old)) + np.nansum(abs(S - S_old))
    # Stop criteria
    if diff <= eps and diff != 0:
        print('Early stopping at step {}, the diff is {}'.format(i+1, diff))
        break
```

Stop criteria are either imposing the maximum number of steps or when the difference of parameter at time t and $t-1$ is less than tolerance level.



4.Repeat the process for 100 times.

5.Evaluation

Evaluate α and β

```
alpha = np.array(alphas)
beta = np.array(betas)

print('For the alpha: the true alpha = {}, mean = {}, variance = {}'.format(true_Z[0][0],np.nanmea
n(alpha), np.nanvar(alpha)))
print('For the beta: the true beta = {}, mean = {}, variance = {}'.format(true_Z[1][0],np.nanmean(
beta), np.nanvar(beta)))
```

```
For the alpha: the true alpha = 0.6, mean = 0.49111817912187017, variance = 0.005399169464344099
For the beta: the true beta = 0.2, mean = 0.5415401478080739, variance = 0.11810709036944107
```

```
alphas[:10]
```

```
[0.4259859546766435,
 0.4259861564413597,
 0.5740137163435277,
 0.42598618959638784,
 0.5740136991027822,
 0.5740137049723929,
 0.4259860550050301,
 0.5740137032514985,
 0.42598623279560804,
 0.4259861644026992]
```

```
betas[:10]
```

```
[0.8461684838117631,
 0.8461685884616391,
 0.1538313455553997,
 0.8461686056582401,
 0.1538313366131052,
 0.15383133965750886,
 0.8461685358493861,
 0.1538313387649288,
 0.8461686280644821,
 0.8461685925909673]
```

Evaluate μ

```
print('The true mean is {}'.format(true_means))
means = np.array(means_s)
#Z_i = 1
print('For the first Gaussian distribution: mean = {}, variance = {}'.format(np.nanmean(means[:,0]
), np.nanvar(means[:,0])))
#Z_i = 2
print('For the second Gaussian distribution: mean = {}, variance = {}'.format(np.nanmean(means[:,1
]), np.nanvar(means[:,1])))
```

The true mean is [15 30]
For the first Gaussian distribution: mean = 23.10218596083503, variance = 53.952352738894575
For the second Gaussian distribution: mean = 21.326501219508685, variance = 53.95229965515634

```
means_s[:10]
```

```
[array([29.61303356, 14.8156572 ]),
 array([29.61303152, 14.81565453]),
 array([14.81565285, 29.61303023]),
 array([29.61303118, 14.8156541 ]),
 array([14.81565263, 29.61303006]),
 array([14.8156527 , 29.61303012]),
 array([29.61303254, 14.81565588]),
 array([14.81565268, 29.6130301 ]),
 array([29.61303075, 14.81565353]),
 array([29.61303144, 14.81565443])]
```

Evalute Variance σ^2

```
print('The true variance is {}'.format(true_S))
S = np.array(Ss)
#Z_i = 1
print('For the first Gaussian distribution: mean = {}, variance = {}'.format(np.nanmean(S[:,0]), n
p.nanvar(S[:,0])))
#Z_i = 2
print('For the second Gaussian distribution: mean = {}, variance = {}'.format(np.nanmean(S[:,1]),
np.nanvar(S[:,1])))
```

The true variance is [20 15]
For the first Gaussian distribution: mean = 17.904541719480065, variance = 21.468705079403673
For the second Gaussian distribution: mean = 16.784445432277, variance = 21.46869904786287

```
Ss[:10]
```

```
[array([22.01164038, 12.67734742]),
 array([22.01165401, 12.67733294]),
 array([12.67732381, 22.0116626 ]),
 array([22.01165625, 12.67733056]),
 array([12.67732257, 22.01166377]),
 array([12.67732299, 22.01166337]),
 array([22.01164716, 12.67734022]),
 array([12.67732287, 22.01166349]),
 array([22.01165917, 12.67732746]),
 array([22.01165455, 12.67733237])]
```

The results shows that, when we have more data, the performance would be much better, and also the variance would be lower. For example, the variance of σ^2 is around 21 when sample size is 100, while around 4 when sample size is 1000.

But in this question, more interesting things happened:

For example, the original parameters are $\alpha = 0.6, \beta = 0.2, \mu_1 = 15, \mu_2 = 30, \sigma_1^2 = 20, \sigma_2^2 = 15$. One set of results from experiment is $\alpha = 0.56, \beta = 0.17, \mu_1 = 15.1, \mu_2 = 30.0, \sigma_1^2 = 19.1, \sigma_2^2 = 15.0$, another one is $\alpha = 0.44, \beta = 0.83, \mu_1 = 29.9, \mu_2 = 15.1, \sigma_1^2 = 15.0, \sigma_2^2 = 19.1$.

This shows that the algorithm does find the optimal solution, but may not as the same order as our initiative. Since in the question, there are only 2 distributions so that 2 groups of parameters. Parameters will switch because the algorithm can not determine the order of these parameters.

So the mean of α is 0.5, the average number of α and $1 - \alpha$, so does β.

The mean of μ equals to the average number of μ_1 and μ_2, so does σ_2.