Question 1 Learning the Parameters

(a) *Derive the M-step update rules for Θ and π by setting the partial derivatives of Equation 6 to zero. Be sure to show your steps.*

Equation 6:

$$F = \sum_{i=1}^{N}\sum_{k=1}^{K} r_k^{(i)}[\log Pr(z^{(i)} = k) + \log p(x^{(i)}|z^{(i)} = k)] + \log p(\pi) + \log p(\Theta)$$

Denote Lagrangian as,

$$L = \sum_{i=1}^{N}\sum_{k=1}^{K} r_k^{(i)} \log Pr(z^{(i)} = k) + \log p(\pi) + \lambda(1 - \sum_{k=1}^{K}\pi_k)$$

To get the derivative of L on π_k,

$$\begin{aligned}\frac{\partial L}{\partial \pi_k} &= \frac{\partial \sum_{i=1}^{N}\sum_{k=1}^{K} r_k^{(i)} \log \pi_k + \log p(\pi) + \lambda(1 - \sum_{k=1}^{K}\pi_k)}{\partial \pi_k} \\ &= \frac{\partial \sum_{i=1}^{N} r_k^{(i)} \log \pi_k + \log \pi_1^{a_1-1}\pi_2^{a_2-1}...\pi_K^{a_K-1} + \lambda(1 - \sum_{k=1}^{K}\pi_k)}{\partial \pi_k} \\ &= \frac{\partial \sum_{i=1}^{N} r_k^{(i)} \log \pi_k + (a_k - 1)\log \pi_k + \lambda(1 - \sum_{k=1}^{K}\pi_k)}{\partial \pi_k} \\ &= \frac{\sum_{i=1}^{N} r_k^{(i)} + a_k - 1}{\pi_k} - \lambda\end{aligned}$$

Let $\frac{\partial F}{\partial \pi_k} = 0$,

$$\lambda = \frac{\sum_{i=1}^{N} r_k^{(i)} + a_k - 1}{\pi_k}$$

Therefore, π_k must be proportional to $\sum_{i=1}^{N} r_k^{(i)} + a_k - 1$,

$$\begin{aligned}\pi_k \leftarrow & \frac{\sum_{i=1}^{N} r_k^{(i)} + a_k - 1}{\sum_{k'=1}^{K}\sum_{i=1}^{N} r_{k'}^{(i)} + a_{k'} - 1} \\ &= \frac{a_k - 1 + \sum_{i=1}^{N} r_k^{(i)}}{N - K + \sum_{k'=1}^{K} a_{k'}} = \frac{a - 1 + \sum_{i=1}^{N} r_k^{(i)}}{K(a-1) + N}\end{aligned}$$

To get the derivative of F on $\theta_{k,j}$,

$$\begin{aligned}\frac{\partial F}{\partial \theta_{k,j}} &= \frac{\partial \sum_{i=1}^{N}\sum_{k=1}^{K} r_k^{(i)} \log p(x^{(i)}|z^{(i)} = k) + \log p(\theta_{k,j})}{\partial \theta_{k,j}} \\ &= \frac{\partial \sum_{i=1}^{N}\sum_{k=1}^{K} r_k^{(i)} \sum_{j=1}^{D}[x_j^{(i)} \log \theta_{k,j} + (1 - x_j^{(i)})\log(1 - \theta_{k,j})] + (a-1)\log\theta_{k,j} + (b-1)\log 1 - \theta_{k,j}}{\partial \theta_{k,j}} \\ &= \frac{\sum_{i=1}^{N} r_k^{(i)} x_j^{(i)} + a - 1}{\theta_{k,j}} - \frac{\sum_{i=1}^{N} r_k^{(i)}(1 - x_j^{(i)}) + b - 1}{1 - \theta_{k,j}}\end{aligned}$$

Let $\frac{\partial F}{\partial \theta_{k,j}} = 0$,

$$\frac{\sum_{i=1}^N r_k^{(i)} x_j^{(i)} + a - 1}{\theta_{k,j}} - \frac{\sum_{i=1}^N r_k^{(i)} (1 - x_j^{(i)}) + b - 1}{1 - \theta_{k,j}} = 0$$

$$(1 - \theta_{k,j})(\sum_{i=1}^N r_k^{(i)} x_j^{(i)} + a - 1) = \theta_{k,j}(\sum_{i=1}^N r_k^{(i)} (1 - x_j^{(i)}) + b - 1)$$

$$(\sum_{i=1}^N r_k^{(i)} + a + b - 2)\theta_{k,j} = \sum_{i=1}^N r_k^{(i)} x_j^{(i)} + a - 1$$

$$\theta_{k,j} \leftarrow \frac{\sum_{i=1}^N r_k^{(i)} x_j^{(i)} + a - 1}{\sum_{i=1}^N r_k^{(i)} + a + b - 2}$$

(b) *Take these formulas and use them to implement the functions Model.update_pi and Model.update_theta in mixture.py. Show the output of running mixture.print_part_1_values().*

```
pi[0] 0.085
pi[1] 0.13
theta[0, 239] 0.6427106227106232
theta[3, 298] 0.46573612495845823
```

Question 2 Posterior Inference

(a) *Derive the rule for computing the posterior probability distribution $p(z|x)$.*

For the whole image,

$$Pr(z = k|x^{(i)}) = \frac{Pr(x^{(i)}|z = k)Pr(z = k)}{\sum_{k'=1}^K Pr(x^{(i)}|z = k')Pr(z = k')}$$

$$= \frac{\prod_{j=1}^D [\theta_{k,j}^{x_j^{(i)}} (1 - \theta_{k,j})^{1 - x_j^{(i)}}]\pi_k}{\sum_{k'=1}^K \prod_{j=1}^D [\theta_{k',j}^{x_j^{(i)}} (1 - \theta_{k',j})^{1 - x_j^{(i)}}]\pi_{k'}}$$

For the partially observed image, the term, of which pixel is observed, should be to m_j power; thus,

$$Pr(z = k|x^{(i)}) = \frac{\prod_{j=1}^D [\theta_{k,j}^{x_j^{(i)}} (1 - \theta_{k,j})^{1 - x_j^{(i)}}]^{m_j}\pi_k}{\sum_{k'=1}^K \prod_{j=1}^D [\theta_{k',j}^{x_j^{(i)}} (1 - \theta_{k',j})^{1 - x_j^{(i)}}]^{m_j}\pi_{k'}}$$

(b) *Implement the method Model.compute_posterior using your solution to the previous question.*

```
R[0, 2] 0.17488951492117283
R[1, 0] 0.6885376761092292
P[0, 183] 0.6516151998131037
P[2, 628] 0.4740801724913301
```

Question 3 Conceptual Questions

(a) *In the code, the default parameters for the beta prior over Θ were $a = b = 2$. If we instead used $a = b = 1$ (which corresponds to a uniform distribution), the MAP learning algorithm would have the problem that it might assign zero probability to images in the test set. Why might this happen?*

If $a = 1$ and $b = 1$, then

$$\theta_{k,j} \leftarrow \frac{\sum_{i=1}^{N} r_k^{(i)} x_j^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}}$$

If a pixel is always 0 in the training set, i.e. we set $x_1^{(i)} = 0$; then,

$$\theta_{k,1} = 0$$

Put the $\theta_{k,1}$ into $Pr(z = k, x^{(i)})$,

$$\begin{aligned} Pr(x^{(i)}|z = k) &= \prod_{j=1}^{D} \theta_{k,j}^{x_j^{(i)}} (1-\theta_{k,j})^{1-x_j^{(i)}} \\ &= [\theta_{k,1}^{x_1^{(i)}} (1-\theta_{k,1})^{1-x_1^{(i)}}]^{m_1} \prod_{j=2}^{D} [\theta_{k,j}^{x_j^{(i)}} (1-\theta_{k,j})^{1-x_j^{(i)}}]^{m_j} \\ &= [0^{x_1^{(i)}} 1^{1-x_1^{(i)}}]^{m_1} \prod_{j=2}^{D} [\theta_{k,j}^{x_j^{(i)}} (1-\theta_{k,j})^{1-x_j^{(i)}}]^{m_j} = 0 \end{aligned}$$

As the probability of images $x^{(i)}$ assigned $z = k$ is proportional to $Pr(x^{(i)}|z = k)$, the MAP learning algorithm would have the problem that it might assign zero probability to images in the test set.
However, if we put the $\theta_{k,1}$ into $Pr(z = k|x^{(i)})$,

$$\begin{aligned} Pr(z = k|x^{(i)}) &= \frac{[\theta_{k,1}^{x_1^{(i)}} (1-\theta_{k,1})^{1-x_1^{(i)}}]^{m_1} \prod_{j=2}^{D} [\theta_{k,j}^{x_j^{(i)}} (1-\theta_{k,j})^{1-x_j^{(i)}}]^{m_j} \pi_k}{\sum_{k'=1}^{K} \prod_{j=1}^{D} [\theta_{k',j}^{x_j^{(i)}} (1-\theta_{k',j})^{1-x_j^{(i)}}]^{m_j} \pi_{k'}} \\ &= \frac{[0^{x_1^{(i)}} 1^{1-x_1^{(i)}}]^{m_1} \prod_{j=2}^{D} [\theta_{k,j}^{x_j^{(i)}} (1-\theta_{k,j})^{1-x_j^{(i)}}]^{m_j} \pi_k}{\sum_{k'=1}^{K} [0^{x_1^{(i)}} 1^{1-x_1^{(i)}}]^{m_1} \prod_{j=2}^{D} [\theta_{k',j}^{x_j^{(i)}} (1-\theta_{k',j})^{1-x_j^{(i)}}]^{m_j} \pi_{k'}} = \frac{0}{\sum_{k'=1}^{K} 0} = NaN \end{aligned}$$

This is another problem.

(b) *The model from Part 2 can still get higher average log probabilities on both the training and test sets, compared with the model from Part 1, even if the number of latent components is set to 10. This is counterintuitive, since the Part 1 model has access to additional information: labels which are part of a true causal explanation of the data (i.e. what digit someone was trying to write). Why do you think the Part 2 model still does better?*

For the method of part 1, they use the true labels as R, so as the values of R are fixed. Then, this method uses the fixed R to get the θ and π. For the method of part 2, they choose the θ and π randomly at first; then, iterate R to get higher average log probabilities. Besides, the clusters extracted from part 2 may be different from 0-9 digits, such as it may cluster 8 and 0 together, which will contribute to higher average log probabilities. If the number of iterations is enough, the method of part 2 will get optimal log probabilities, which should be higher than the probabilities from the method of part 1.

Question 3 continued on next page...

(c) *The function print_log_probs_by_digit_class computes the average log-probabilities for different digit classes in both the training and test sets. In both cases, images of 1's are assigned far higher log-probability than images of 8's. Does this mean the model thinks 1's are far more common than 8's? I.e., if you sample from its distribution, will it generate far more 1's than 8's? Why or why not?*

This method uses Multinomial distribution to generate class and uses a collection of Bernoulli distribution to generate images. Thus, the model treats the parameter π_1 and π_8 as the probabilities to generate 1's and 8's. As shown in the Equation of π_k,

$$\pi_k \leftarrow \frac{a - 1 + \sum_{i=1}^{N} r_k^{(i)}}{K(a-1) + N}$$

π_k is decided by $r_k^{(i)}$.
However, some digits, like 8's, is hard to be detected, as it may be considered as 0's or 9's, which means the model has less confidence to recognize 8's, so as to make the value of $\sum_{i-1}^{N} r_8$ smaller than $\sum_{i-1}^{N} r_1$. As a result, the value of π_8 is smaller than π_1, which will make the model computes higher the average log-probabilities for 1's than 8's and, thus, sample more 1's than 8's from its distribution.