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 $\theta_{k,j}$,

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

Question 1 [(a)] continued on next page...

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().*

(c)

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.*

(c) *Implement the method Model.posterior_predictive_means, which computes the posterior predictive means of the missing pixels given the observed ones.*

Question 2 continued on next page...

(d)

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?*

(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?*

(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?*