CSC311 Assignment3

Peiqing Yu 2020/11/4 1003769004

1. (a). First, divide the summation to two sets $E = \{i : h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}$ and its complement $E^c = \{i : h_t(\mathbf{x}^{(i)}) = t^{(i)}\}$.

Then we have $\dfrac{\sum_{i \in E} w_i}{\sum_{i=1}^{N} w_i} = err_t$

We can also use the equivalent weight update rule:

$$w_i' \leftarrow w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}\right)$$

Thus,

$$err_t' = \frac{\sum_{i=1}^{N} w_i' \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i'}$$

$$= \frac{\sum_{i \in E} w_i' \overset{1}{\mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}} + \sum_{i \in E^c} w_i' \overset{0}{\mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}}{\sum_{i=1}^{N} w_i}$$

$$= \frac{\sum_{i \in E} w_i' \cdot 1 + \sum_{i \in E^c} w_i' \cdot 0}{\sum_{i=1}^{N} w_i'}$$

$$= \frac{\sum_{i \in E} w_i'}{\sum_{i=1}^{N} w_i'}$$

$$= \frac{\sum_{i \in E} w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}\right)}{\sum_{i=1}^{N} w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}\right)}$$

$$= \frac{\sum_{i \in E} w_i \exp\left(2\alpha_t \overset{1}{\mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}\right)}{\sum_{i \in E} w_i \exp\left(2\alpha_t \overset{1}{\mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}\right) + \sum_{i \in E^c} w_i \exp\left(2\alpha_t \overset{0}{\mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}\right)}$$

$$= \frac{\sum_{i \in E} w_i \exp(2\alpha_t)}{\sum_{i \in E} w_i \exp(2\alpha_t) + \sum_{i \in E^c} w_i}$$

\# Now sub in $\alpha_t = \frac{1}{2} \log \frac{1 - err_t}{err_t}$

$$= \frac{\sum_{i \in E} w_i \left(\frac{1 - err_t}{err_t}\right)}{\sum_{i \in E} w_i \left(\frac{1 - err_t}{err_t}\right) + \sum_{i \in E^c} w_i}$$

\# Sub in $err_t = \dfrac{\sum_{i \in E} w_i}{\sum_{i=1}^{N} w_i}$

$$= \frac{\left(\frac{\sum_{i=1}^{N} w_i}{\sum_{i \in E} w_i} - 1\right) \sum_{i \in E} w_i}{\left(\frac{\sum_{i=1}^{N} w_i}{\sum_{i \in E} w_i} - 1\right) \sum_{i \in E} w_i + \sum_{i \in E^c} w_i}$$

$$= \frac{\sum_{i=1}^{N} w_i - \sum_{i \in E} w_i}{\sum_{i=1}^{N} w_i - \sum_{i \in E} w_i + \sum_{i \in E^c} w_i}$$

$$= \frac{\sum_{i \in E^c} w_i}{2 \sum_{i \in E^c} w_i}$$

$$= \frac{1}{2}$$

This implies that at each iteration the sum of weights to be changes is exactly

1/2. Since the number of misclassified training data decreases and the weights

to be changes are fixed, this allows the decrease in loss.

(b).

$$w_i \exp(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\})$$

$$= w_i \exp(2\alpha_t \cdot \frac{1}{2}(1 - h(\mathbf{x}^{(i)}) \cdot t^{(i)}))$$

$$= w_i \exp(\alpha_t - \alpha_t h(\mathbf{x}^{(i)}) t^{(i)})$$

$$= w_i e^{\alpha_t} \cdot \exp(-\alpha_t h(\mathbf{x}^{(i)}) t^{(i)})$$

$$\propto w_i \cdot \exp(-\alpha_t h(\mathbf{x}^{(i)}) t^{(i)})$$

Thus, the constant factor is e^{α_t}, which is constant across all weights

Q2 (a).

$$p(D \mid \boldsymbol{\theta}, \boldsymbol{\pi})$$

$$= \prod_{i=1}^{N} P(\mathbf{x}^{(i)}, \mathbf{t}^{(i)} \mid \boldsymbol{\theta}, \boldsymbol{\pi})$$

$$= \prod_{i=1}^{N} p(\mathbf{t}^{(i)} \mid \boldsymbol{\theta}, \boldsymbol{\pi}) \, p(\mathbf{x}^{(i)} \mid \mathbf{t}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\pi})$$

$$= \prod_{i=1}^{N} p(\mathbf{t}^{(i)} \mid \boldsymbol{\pi}) \prod_{j=1}^{784} p(x_j^{(i)} \mid \boldsymbol{\theta}_{jc^{(i)}}, \mathbf{t}^{(i)})$$

$$l(\boldsymbol{\theta}, \boldsymbol{\pi}) = \underbrace{\sum_{i=1}^{N} \log p(\mathbf{t}^{(i)} \mid \boldsymbol{\pi})}_{①} + \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{784} \log p(x_j^{(i)} \mid \boldsymbol{\theta}_{jc^{(i)}}, \mathbf{t}^{(i)})}_{②}$$

Note that ① doesn't depend on θ, and ② doesn't depend on π

To derive MLE for θ and π, we can only consider ②, ① respectively.

1) Derive $\hat{\theta}_{MLE}$

$$l_1(\boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{j=1}^{784} \log p(x_j^{(i)} \mid \boldsymbol{\theta}_{jc^{(i)}}, \mathbf{t}^{(i)})$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{784} \log \left[\boldsymbol{\theta}_{jc^{(i)}}^{x_j^{(i)}} \cdot (1 - \boldsymbol{\theta}_{jc^{(i)}})^{1 - x_j^{(i)}} \right]$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{784} \left[x_j^{(i)} \log \theta_{jc^{(i)}} + (1 - x_j^{(i)}) \log(1 - \theta_{jc^{(i)}}) \right]$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{784} \left[x_j^{(i)} \log \theta_{jc^{(i)}} + (1 - x_j^{(i)}) \log(1 - \theta_{jc^{(i)}}) \right]$$

since we take derivative wrt. θ_{jc}, we can get rid of $\sum_{j=1}^{784}$ and $c^{(i)}$

$$\frac{\partial l_1(\theta)}{\partial \theta_{jc}} = \sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} \frac{x_j^{(i)}}{\theta_{jc}} - \frac{1 - x_j^{(i)}}{1 - \theta_{jc}} = 0$$

$$\theta_{jc}(1-\theta_{jc}) \sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} \frac{x_j^{(i)}}{\theta_{jc}} - \frac{1 - x_j^{(i)}}{1 - \theta_{jc}} = 0 \cdot \theta_{jc}(1-\theta_{jc})$$

$$\sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} (x_j^{(i)} - \cancel{x_j^{(i)}\theta_{jc}} - \theta_{jc} + \cancel{x_j^{(i)}\theta_{jc}}) = 0$$

$$\sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} x_j^{(i)} = \sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} \theta_{jc}$$

$$\theta_{jc} = \frac{\sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} x_j^{(i)}}{\sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\}}$$

Thus we have $\hat{\theta}_{MLE}$ with entry $\hat{\theta}_{jc} = \frac{\sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} x_j^{(i)}}{\sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\}}$
where $j \in \{1, 2, \cdots, 784\}$, and $c \in \{0, 1, \cdots, 9\}$
We assure that $\hat{\theta_{jc}}$ is maximized by checking l_1''

Note that θ hat is the measure of the counts of x_j feature in class c over the total number of class c input data

2) Derive $\hat{\pi}_{MLE}$

$$l_2(\pi) = \sum_{i=1}^{N} \log P(t^{(i)} | \pi)$$

$$= \sum_{i=1}^{N} \log \left(\prod_{j=0}^{9} \pi_j^{t_j^{(i)}} \right)$$

$$= \sum_{i=1}^{N} \log \left(\prod_{j=0}^{8} \pi_j^{t_j^{(i)}} \cdot \pi_9^{t_9^{(i)}} \right)$$

$$= \sum_{i=1}^{N} \log \left(\prod_{j=0}^{8} \pi_j^{t_j^{(i)}} \cdot \left(1 - \sum_{k=0}^{8} \pi_k\right)^{t_9^{(i)}} \right)$$

$$= \sum_{i=1}^{N} \left[\sum_{j=0}^{8} t_j^{(i)} \log \pi_j + t_9^{(i)} \log\left(1 - \sum_{k=0}^{8} \pi_k\right) \right]$$

\# since we take derivative wrt. π_j, we can get rid of $\sum_{j=0}^{8}$

$$\frac{\partial l_2(\pi)}{\partial \pi_j} = \sum_{i=1}^{N} \frac{t_j^{(i)}}{\pi_j} - \frac{t_9^{(i)}}{1-\sum_{k=0}^{8}\pi_k} = 0$$

$$\sum_{i=1}^{N} \frac{t_j^{(i)}}{\pi_j} - \frac{t_9^{(i)}}{\pi_9} = 0$$

$$\frac{\sum_{i=1}^{N} t_j^{(i)}}{\pi_j} = \frac{\sum_{i=1}^{N} t_9^{(i)}}{\pi_9}$$

$$\frac{\pi_j}{\pi_9} = \frac{\sum_{i=1}^{N} t_j^{(i)}}{\sum_{i=1}^{N} t_9^{(i)}}$$

We assure that $\frac{\hat{\pi_j}}{\hat{\pi_9}}$ is maximized by checking l_2''

since we have $\frac{\hat{\pi_0}}{\hat{\pi_9}} + \frac{\hat{\pi_1}}{\hat{\pi_9}} + \cdots + \frac{\hat{\pi_9}}{\hat{\pi_9}} = \frac{\hat{\pi_0}+\hat{\pi_1}+\cdots+\hat{\pi_9}}{\hat{\pi_9}} = \frac{1}{\hat{\pi_9}}$

$$\frac{\sum_{i=1}^{N} t_0^{(i)}}{\sum_{i=1}^{N} t_9^{(i)}} + \frac{\sum_{i=1}^{N} t_1^{(i)}}{\sum_{i=1}^{N} t_9^{(i)}} + \cdots + \frac{\sum_{i=1}^{N} t_9^{(i)}}{\sum_{i=1}^{N} t_9^{(i)}} = \frac{1}{\hat{\pi_9}}$$

$$\frac{\sum_{i=1}^{N} t_0^{(i)}+t_1^{(i)}+\cdots+t_9^{(i)}}{\sum_{i=1}^{N} t_9^{(i)}} = \frac{1}{\hat{\pi_9}}$$

$$\hat{\pi_9} = \frac{\sum_{i=1}^{N} t_9^{(i)}}{N}$$

Note that N is the total number of data points, as t is 1-of-10 encoded vector.

So if we change to seperate $\pi_0, \pi_1, \cdots, \pi_8$ respectively, we can have $\hat{\pi}_{MLE}$ with entry $\hat{\pi_j} = \frac{\sum_{i=1}^{N} t_j^{(i)}}{N}$, where $j \in \{0,1,2,\cdots,9\}$

Note that π_j hat is the number of class j over the total number of input data.

```python
def train_mle_estimator(train_images, train_labels):
    """ Inputs: train_images, train_labels
        Returns the MLE estimators theta_mle and pi_mle"""

    # YOU NEED TO WRITE THIS PART

    # compute the counts of each class c in train_labels
    t_sum = np.sum(train_labels, axis=0)
    theta_mle = (train_images.T).dot(train_labels)/t_sum
    pi_mle = t_sum/train_labels.shape[0]
    return theta_mle, pi_mle
```

(b) $$p(t|x,\theta,\pi) = \frac{p(t_c=1|\pi)\, p(x|t,\theta,\pi)}{\sum_{k=0}^{9} p(t_k=1)\, p(x|t,\theta,\pi)}$$

$$= \frac{p(t_c|\pi) \prod_{j=1}^{784} p(x_j|\theta_{jc}, t_c)}{\sum_{k=0}^{9} p(t_k=1) \prod_{j=1}^{784} p(x_j|\theta_{jc}, c=k)}$$

$$= \frac{\pi_c \prod_{j=1}^{784} \theta_{jc}^{x_j} (1-\theta_{jc})^{1-x_j}}{\sum_{k=0}^{9} \pi_k \prod_{j=1}^{784} \theta_{jk}^{x_j} (1-\theta_{jk})^{1-x_j}}$$

$$\log p(t|x,\theta,\pi) = \log \pi_c + \sum_{j=1}^{784} x_j \log \theta_{jc} + \sum_{j=1}^{784} (1-x_j) \log(1-\theta_{jc})$$
$$- \log\left[\sum_{k=0}^{9} \exp\left(\log \pi_k + \sum_{j=1}^{784} x_j \log \theta_{jk} + \sum_{j=1}^{784} (1-x_j) \log(1-\theta_{jc})\right)\right]$$

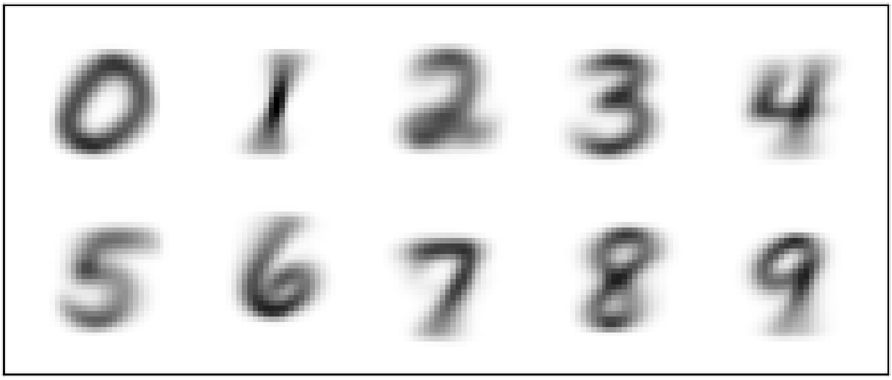
```
def log_likelihood(images, theta, pi):
    """ Inputs: images, theta, pi
        Returns the matrix 'log_like' of loglikehoods over the input images where
    log_like[i,c] = log p (c |x^(i), theta, pi) using the estimators theta and pi.
    log_like is a matrix of num of images x num of classes
    Note that log likelihood is not only for c^(i), it is for all possible c's."""

    # YOU NEED TO WRITE THIS PART
    first = np.log(pi)+np.dot(images, np.log(theta))+np.dot((1-images), np.log(1-theta))
    # note to import e from math module
    power = np.array(e)**first
    second = np.log((np.sum(power, axis=1)).reshape(first.shape[0], 1))
    log_like = first - second
    return log_like
```

(c) The average log-likelihood per data point by fitting θ and π using training set with MLE is nan. The result goes wrong because a lot of the entries of θ_{MLE} is 0, which cause log 0 error. This is due to the data sparsity of the input x and t.

```
Average log-likelihood for MLE is  nan
```

(d) MLE estimator θ hat for all of the 10 classes



We can observe that the MLE estimators look very similar to the real hard-

Writen digits.

(e) $$p(\theta_{jc}, a=3, b=3) = \frac{\Gamma(6)}{\Gamma(3)+\Gamma(3)} \theta_{jc}^{2} (1-\theta_{jc})^{2}$$

$$\propto \theta_{jc}^{2} (1-\theta_{jc})^{2}$$

$$\hat{\theta}_{jc_{MAP}} = \arg\max_{\theta} p(\theta | D)$$

$$p(\theta | D) = p(\theta, D)$$

$$= p(\theta) p(D | \theta)$$

$$= \theta_{jc}^{2} (1-\theta_{jc})^{2} \prod_{i=1}^{N} p(c^{(i)} | \pi) \prod_{j=1}^{784} p(x_j^{(i)} | c, \theta)$$

$$= \theta_{jc}^{2} (1-\theta_{jc})^{2} \prod_{i=1}^{N} \pi_c^{(i)} \prod_{j=1}^{784} \theta_{jc^{(i)}}^{x_j^{(i)}} (1-\theta_{jc^{(i)}})^{1-x_j^{(i)}}$$

$$l = \log p(\theta | D) = 2\log\theta_{jc} + 2\log(1-\theta_{jc}) + \sum_{i=1}^{N}\left[\log \pi_c^{(i)} + \sum_{j=1}^{784} x_j^{(i)} \log \theta_{jc}^{(i)} + \sum_{j=1}^{784} (1-x_j^{(i)}) \log(1-\theta_{jc}^{(i)})\right]$$

since we take derivative wrt. θ_{jc}, we can get rid of $\sum_{j=1}^{784}$ and $c^{(i)}$

$$\frac{\partial l}{\partial \theta_{jc}} = \frac{2}{\theta_{jc}} - \frac{2}{1-\theta_{jc}} + \sum_{i=1}^{N}\left(\frac{x_j^{(i)}}{\theta_{jc^{(i)}}} - \frac{1-x_j^{(i)}}{1-\theta_{jc^{(i)}}}\right) = 0$$

$$\frac{2}{\theta_{jc}} - \frac{2}{1-\theta_{jc}} + \sum_{i=1}^{N}\left[\mathbb{I}\{c^{(i)}=c\}\left(\frac{x_j^{(i)}}{\theta_{jc}} - \frac{1-x_j^{(i)}}{1-\theta_{jc}}\right)\right] = 0$$

$$2 - 2\theta_{jc} - 2\theta_{jc} + \sum_{i=1}^{N} \mathbb{I}\{c^{(i)}=c\} x_j^{(i)} - \sum_{i=1}^{N} \mathbb{I}\{c^{(i)}=c\} \theta_{jc} = 0 \cdot \theta_{jc}(1-\theta_{jc})$$

$$2 + \sum_{i=1}^{N} \mathbb{I}\{c^{(i)}=c\} x_j^{(i)} = \left(4 + \sum_{i=1}^{N} \mathbb{I}\{c^{(i)}=c\}\right) \theta_{jc}$$

$$\hat{\theta}_{jc} = \frac{2 + \sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} x_j^{(i)}}{4 + \sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\}}$$

We assure that $\hat{\theta}_{jc}$ is maximized by checking l''

Thus we have $\hat{\theta}_{MAP}$ with entry $\hat{\theta}_{jc} = \frac{2 + \sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\} x_j^{(i)}}{4 + \sum_{i=1}^{N} \mathbb{I}\{c^{(i)} = c\}}$

where $j \in \{1, 2, \cdots, 784\}$ and $c \in \{0, 1, \cdots, 9\}$

(f) The average log-likelihood per data point by fitting θ and π using training set with MAP is approximately -3.357. The train accuracy for MAP is approximately 0.835. The test accuracy for MAP is 0.816.

```
Average log-likelihood for MAP is  -3.3570631378602904
Training accuracy for MAP is  0.8352166666666667
Test accuracy for MAP is  0.816
```

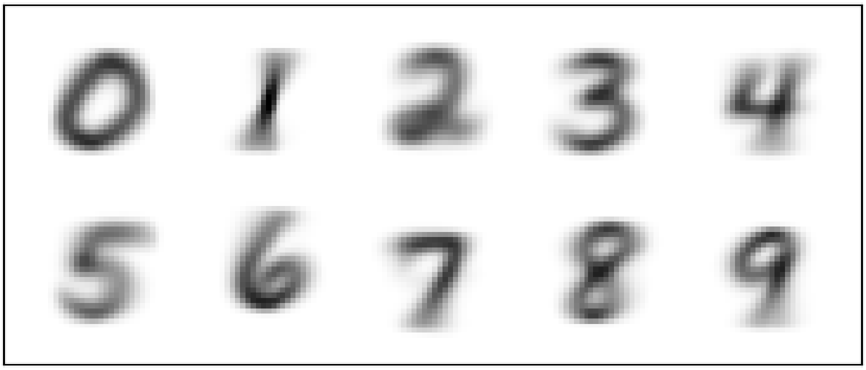
```
def predict(log_like):
    """ Inputs: matrix of log likelihoods
    Returns the predictions based on log likelihood values"""

    # YOU NEED TO WRITE THIS PART
    # create a zero matrix with the same size of log matrix
    predictions = np.zeros((log_like.shape[0], log_like.shape[1]))
    # compute the index of each training data that with highest log likelihood
    idx = log_like.argmax(axis=1)
    predictions[np.arange(log_like.shape[0]), idx] = 1
    return predictions


def accuracy(log_like, labels):
    """ Inputs: matrix of log likelihoods and 1-of-K labels
    Returns the accuracy based on predictions from log likelihood values"""

    # YOU NEED TO WRITE THIS PART
    pred = predict(log_like).argmax(axis=1)
    accuracy = np.mean(pred == labels.argmax(axis=1))
    return accuracy
```

(g) MAP estimator θ hat for all of the 10 classes



We can observe that the MAP estimators look very similar to the real hard-Writtern digits.

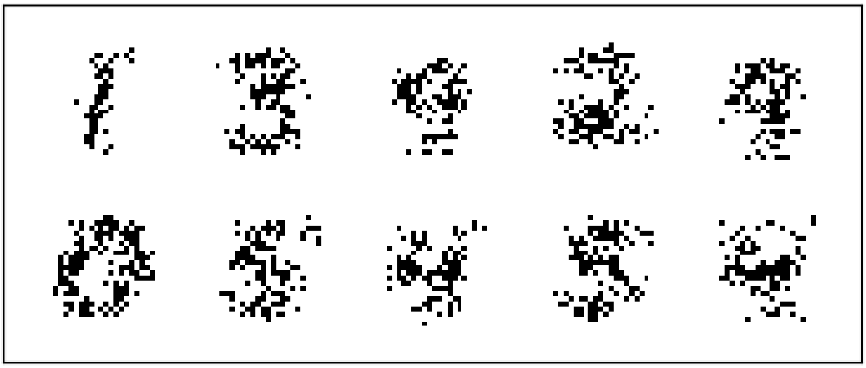
Q3 (a). True. Naïve Bayes model assumes that xi and xj are conditionally independent given the class c.

(b). False.

After marginalization over c, $\sum_{c'} p(x_i, c') = p(x_i)$, $\sum_{c'} p(x_j, c') = p(x_j)$
However, $p(x_i, x_j)$ not necessarily equal to $p(x_i)p(x_j)$, because naive bayes assumption does not include x_i and x_j are independent (for $i \neq j$). Thus the statement is False

(c) Randomly sample 10 plots from $p(\mathbf{x}|\hat{\theta}, \hat{\pi})$.We can observe that the generated samples are very similar to the class we randomly selected.

```
The randomly selected samples are  [1 3 9 2 9 0 5 4 5 4]
```



```python
def image_sampler(theta, pi, num_images):
    """ Inputs: parameters theta and pi, and number of images to sample
    Returns the sampled images"""

    # YOU NEED TO WRITE THIS PART
    # sample random variable c
    c = np.random.choice(10, num_images, p=pi)
    print("The randomly selected samples are ", c)
    # corresponding theta_jc in a matrix with size D * num_images
    theta_jc = theta[:, c]
    sampled_images = (np.random.binomial(1, theta_jc)).T
    return sampled_images
```