1a) $L(\theta) = \theta_1^{\sum_{i=1}^{N} x_1^{(i)}} \theta_2^{\sum_{i=1}^{N} x_2^{(i)}} \cdots \theta_K^{\sum_{i=1}^{N} x_K^{(i)}}$

$= \theta_1^{N_1} \cdot \theta_2^{N_2} \cdot \ldots \cdot \theta_K^{N_K}$

$= \prod_{k=1}^{K-1} \theta_k^{N_k} \cdot (1 - \sum_{k=1}^{K-1} \theta_k)^{N_K}$

Then $l(\theta) = \sum_{k=1}^{K-1} \log \theta_k^{N_k} + \log (1 - \sum_{i=1}^{K-1} \theta_i)^{N_K}$

$= \sum_{k=1}^{K-1} N_k \log \theta_k + N_K \log (1 - \sum_{i=1}^{K-1} \theta_i)$

$= \sum_{k=1}^{K-1} N_k \log \theta_k + (N - \sum_{k=1}^{K-1} N_k) \log (1 - \sum_{i=1}^{K-1} \theta_i)$

Set $\partial / \partial \theta_k \, l(\theta) = N_k / \theta_k - (N - \sum_{k=1}^{K-1} N_k) / 1 - \sum_{i=1}^{K-1} \theta_i = 0$

Then $\theta_k = \dfrac{N_k (1 - \sum_{k=1}^{K-1} \theta_k)}{(N - \sum_{k=1}^{K-1} N_k)}$

$= \dfrac{N_k}{N} \cdot \dfrac{(1 - \sum_{k=1}^{K-1} \theta_k)}{(1 - \sum_{i=1}^{K-1} N_k / N)}$

$= N_k / N \cdot \dfrac{\theta_K}{N_K / N}$ for all $k \in \{1, \ldots, K-1\}$

$= \dfrac{N_k \theta_K}{N_K}$

Since $\sum_{i=1}^{K} \theta_i = 1$

Then $\dfrac{\theta_K}{N_K} \sum_{i=1}^{K-1} N_i + \theta_K = 1$

$\theta_K \left(\dfrac{\sum_{i=1}^{K-1} N_i}{N_K} + 1 \right) = 1$

$\theta_K \dfrac{N_K + \sum_{i=1}^{K-1} N_i}{N_K} = 1$

$\theta_K \dfrac{N}{N_K} = 1$

$\theta_K = \dfrac{N_K}{N}$

So for $k \in \{1, \ldots, K-1\}$, $\theta_k = \dfrac{N_k \frac{N_K}{N}}{N_K} = \dfrac{N_k}{N}$

So $\forall k \in \{1, \ldots, K\}$, $\theta_k = \dfrac{N_k}{N}$

1b) $p(\theta \mid D) \propto p(\theta)\, p(D \mid \theta)$

$$\propto \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \prod_{i=1}^{k} \theta_i^{\sum_{j=1}^{N} x_i^{(j)}}$$

$$\propto \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \prod_{i=1}^{k} \theta_i^{N_i}$$

$$= \prod_{i=1}^{k} \theta_i^{\alpha_i + N_i - 1}$$

So $p(\theta \mid D) \sim \text{Dirichlet}(\alpha_1 + N_1, \ldots, \alpha_k + N_k)$

1 c) For $i \in \{1, \ldots, k\}$

$$\hat{\theta}_{i\,map} = \underset{\theta_i}{\operatorname{argmax}}\; p(\theta \mid D)$$

$$= \underset{\theta_i}{\operatorname{argmax}}\; p(\theta) + p(D \mid \theta)$$

Since $p(\theta) \propto \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1}$

$$p(D \mid \theta) = \prod_{k=1}^{k} \theta_k^{N_k}$$

So $\hat{\theta}_{i\,map} = \underset{\theta_i}{\operatorname{argmax}} \prod_{k=1}^{k} \theta_k^{\alpha_k + N_k - 1}$

$$= \underset{\theta_i}{\operatorname{argmax}} \sum_{i=1}^{k} (\alpha_i + N_i - 1) \log \theta_i$$

Set $\theta_k = 1 - \sum_{z=1}^{k-1} \theta_j$

$$\hat{\theta}_{i\,map} = \underset{\theta_i}{\operatorname{argmax}} \sum_{j=1}^{k-1} (\alpha_j + N_j - 1) \log \theta_j + (\alpha_k + N_k - 1) \log 1 - \sum_{z=1}^{k-1} \theta_z$$

Set $d/d\theta_i \sum_{j=1}^{k-1} (\alpha_j + N_j - 1) \log \theta_j + (\alpha_k + N_k - 1) \log 1 - \sum_{z=1}^{k-1} \theta_z = 0$

$$\alpha_i + N_i - 1 / \theta_i - \alpha_k + N_k - 1 / \theta_k = 0$$

$$\Rightarrow \theta_i = \frac{\alpha_i + N_i - 1}{\alpha_k + N_k - 1} \cdot \theta_k$$

Since $\sum_{j=1}^{k-1} \theta_j + \theta_k = 1$

So $\dfrac{\sum_{j=1}^{k-1} \alpha_j + \sum_{j=1}^{k-1} N_j - k - 1}{\alpha_k + N_k - 1} \theta_k + \theta_k = 1$

$$\theta_k = \frac{\alpha_k + N_k - 1}{\sum_{j=1}^{k} \alpha_j + N - k}$$

Substitute θ_k into θ_i, $\theta_i = \dfrac{\alpha_i + N_i - 1}{\sum_{j=1}^{k} \alpha_j + N - k}$

So $\forall i \in \{1, \ldots, k\}$

$$\hat{\theta}_{i\,map} = \frac{N_i - 1 + \alpha_i}{N - k + \sum_{j=1}^{k} \alpha_j}$$

1 d) $\theta_{kpred} = P(x_k^{N+1} = 1 \mid D)$

$$= \int p(\theta \mid D)\, P(x_k^{N+1} = 1 \mid \theta)\, d\theta$$

$$= \int \text{Dirichlet}(a_1 + N_1, \ldots, a_k + N_k) \prod_{i=1}^{k} \theta_i^{x_i^{N+1}}\, d\theta$$

$$= \int \text{Dirichlet}(a_1 + N_1, \ldots, a_k + N_k)\, \theta_k\, d\theta$$

$$= E(\theta_k) = \frac{a_k + N_k}{\sum_{j=1}^{k} a_j + N_j} = \frac{a_k + N_k}{\sum_{j=1}^{k} a_j + N}$$

so $E[\theta_k \mid D] = \dfrac{N_k + a_k}{N + \sum_{j=1}^{k} a_j}$

2a) $p(x) = \sum_{i=1}^{k} p(x \mid Y=i)\, p(Y=i)$

$$= \sum_{i=1}^{k} \left(\prod_{d=1}^{D} 2\pi\sigma_d^2\right)^{-1/2} \exp\left\{-\sum_{d=1}^{D} \frac{1}{2\sigma_d^2} (x_d - \mu_{id})^2\right\} \cdot \alpha_i$$

$$= \left(\prod_{d=1}^{D} 2\pi\sigma_d^2\right)^{-1/2} \sum_{i=1}^{k} \exp\left\{-\sum_{d=1}^{D} \frac{1}{2\sigma_d^2} (x_d - \mu_{id})^2\right\} \cdot \alpha_i$$

$$p(Y=k \mid x) = p(x \mid Y=k) \cdot p(Y=k) \,/\, p(x)$$

$$= \frac{\exp\left\{-\sum_{d=1}^{D} \frac{1}{2\sigma_d^2} (x_d - \mu_{kd})^2\right\} \cdot \alpha_k}{\sum_{i=1}^{k} \exp\left\{-\sum_{d=1}^{D} \frac{1}{2\sigma_d^2} (x_d - \mu_{id})^2\right\} \cdot \alpha_i}$$

2b) $l(\theta) = \sum_{i=1}^{N} \log p(Y^{(i)}, x^{(i)})$

$$= \sum_{i=1}^{N} \log p(Y^{(i)})\, p(x^{(i)} \mid Y^{(i)})$$

$$= \sum_{i=1}^{N} \log p(Y^{(i)}) + \log p(x^{(i)} \mid Y^{(i)})$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{k} \log p(Y^{(i)}=k)\, \mathbb{1}(Y^{(i)}=k) + \log p(x^{(i)} \mid Y^{(i)}=k)\, \mathbb{1}(Y^{(i)}=k)$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{k} \left(\log \alpha_k + \log\left(\prod_{d=1}^{D} 2\pi\sigma_d^2\right)^{-1/2} \exp\left\{-\sum_{d=1}^{D} 1/2\sigma_d^2 \,(x_d^{(i)} - \mu_{kd})^2\right\}\right) \mathbb{1}(Y^{(i)}=k)$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{k} \left(\log \alpha_k - 1/2 \sum_{d=1}^{D} \log 2\pi\sigma_d^2 - \sum_{d=1}^{D} 1/2\sigma_d^2 \,(x_d^{(i)} - \mu_{kd})^2\right) \mathbb{1}(Y^{(i)}=k)$$

2C) $\partial / \partial \sigma_d^2 \, l(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{k} \left(-1/2 \cdot \frac{2\pi}{2\pi \sigma_d^2} + 1/2\sigma_d^4 \, (x_d^{(i)} - \mu_{kd})^2 \right) \mathbb{1}(y^{(i)} = k)$

Set $\partial / \partial \sigma_d^2 \, l(\theta) = 0$

Then $1/\sigma_d^2 \sum_{i=1}^{N} \sum_{k=1}^{K} 1/2 \, \mathbb{1}(y^{(i)} = k) = 1/2\sigma_d^4 \sum_{i=1}^{N} \sum_{k=1}^{K} (x_d^{(i)} - \mu_{kd})^2 \, \mathbb{1}(y^{(i)} = k)$

$$\sigma_d^2 = \frac{\sum_{i=1}^{N} \sum_{k=1}^{K} (x_d^{(i)} - \mu_{kd})^2 \, \mathbb{1}(y^{(i)} = k)}{\sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}(y^{(i)} = k)}$$

$$\sigma_d^2 = \frac{\sum_{i=1}^{N} \sum_{k=1}^{K} (x_d^{(i)} - \mu_{kd})^2 \, \mathbb{1}(y^{(i)} = k)}{N}$$

$\partial / \partial \mu_{kd} \, l(\theta) = \sum_{i=1}^{N} 1/\sigma_d^2 \, (x_d^{(i)} - \mu_{kd}) \, \mathbb{1}(y^{(i)} = k)$

Set $\partial / \partial \mu_{kd} \, l(\theta) = 0$

$\sum_{i=1}^{N} 1/\sigma_d^2 \, (x_d^{(i)} - \mu_{kd}) \, \mathbb{1}(y^{(i)} = k) = 0$

$\sum_{i=1}^{N} (1/\sigma_d^2) \, x_d^{(i)} \cdot \mathbb{1}(y^{(i)} = k) = \mu_{kd} \sum_{i=1}^{N} (1/\sigma_d^2) \, \mathbb{1}(y^{(i)} = k)$

$\sum_{i=1}^{N} x_d^{(i)} \, \mathbb{1}(y^{(i)} = k) = \mu_{kd} N_k$

$$\mu_{kd} = \frac{\sum_{i=1}^{N} x_d^{(i)} \, \mathbb{1}(y^{(i)} = k)}{N_k}$$

So $\mu_{kd} = \frac{\sum_{i=1}^{N} x_d^{(i)} \, \mathbb{1}(y^{(i)} = k)}{N_k}$

$\sigma_d^2 = \frac{\sum_{i=1}^{N} \sum_{k=1}^{K} (x_d^{(i)} - \mu_{kd})^2 \, \mathbb{1}(y^{(i)} = k)}{N}$

3 a)

```python
def compute_mean_mles(train_data, train_labels):
    '''
    Compute the mean estimate for each digit class. You may iterate over
    the possible digits (0 to 9), but otherwise make sure that your code
    is vectorized.

    Arguments
        train_data: size N x 64 numpy array with the images
        train_labels: size N numpy array with corresponding labels

    Returns
        means: size 10 x 64 numpy array with the ith row corresponding
               to the mean estimate for digit class i
    '''
    # Initialize array to store means
    means = np.zeros((10, 64))
    for t in range(10):
        k = t  # initialize the label
        td = train_data.copy()
        tl = train_labels.copy()
        tl[train_labels != k] = 0
        tl[train_labels == k] = 1
        tl = np.matrix(tl)
        nk = tl.sum()  # find the total number of label = k in training set
        m = (tl * td) / nk
        means[k, :] = m
    return means
```

3 a)

```python
def compute_sigma_mles(train_data, train_labels):
    '''
    Compute the covariance estimate for each digit class. You may iterate over
    the possible digits (0 to 9), but otherwise make sure that your code
    is vectorized.

    Arguments
        train_data: size N x 64 numpy array with the images
        train_labels: size N numpy array with corresponding labels

    Returns
        covariances: size 10 x 64 x 64 numpy array with the ith row corresponding
                to the covariance matrix estimate for digit class i
    '''
    # Initialize array to store covariances
    covariances = np.zeros((10, 64, 64))
    # == YOUR CODE GOES HERE ==
    for t in range(10):
        k = t
        mean_k = compute_mean_mles(train_data, train_labels)[k, :]
        td = train_data.copy()
        tl = train_labels.copy()
        tl[train_labels != k] = 0
        tl[train_labels == k] = 1
        ind = np.where(tl == 1)[0]  # Find the index of training label k
        nk = ind.shape[0]  # Find Nk
        td = td[ind]  # Find the data whose training label is k
        c = np.dot((td - mean_k).T, (td - mean_k))/nk \
            + 0.01 * np.identity(64)
        covariances[k, :, :] = c
    # ====
    return covariances
```

3 a)

```python
def generative_likelihood(digits, means, covariances):
    '''
    Compute the generative log-likelihood log p(x|t). You may iterate over
    the possible digits (0 to 9), but otherwise make sure that your code
    is vectorized.

    Arguments
        digits: size N x 64 numpy array with the images
        means: size 10 x 64 numpy array with the 10 class means
        covariances: size 10 x 64 x 64 numpy array with the 10 class covariances

    Returns
        likelihoods: size N x 10 numpy array with the ith row corresponding
                to logp(x^(i) | t) for t in {0, ..., 9}
    '''
    N = digits.shape[0]
    likelihoods = np.zeros((N, 10))
    d = 64
    # == YOUR CODE GOES HERE ==
    for t in range(10):
        m = means[t, :]
        c = covariances[t, :, :]
        x = digits
        cinv = np.linalg.inv(c)  # compute the inverse of covariance matrix
        p = -(d/2)*np.log(2*math.pi) - (1/2)*np.log(np.linalg.det(c)) \
            - 0.5 * np.diagonal(np.dot(np.dot(x-m, cinv), (x-m).T))
        likelihoods[:, t] = p
    # ====
    return likelihoods
```

3 a)

```python
def conditional_likelihood(digits, means, covariances):
    '''
    Compute the generative log-likelihood log p(t|x). Make sure that your code
    is vectorized.

    Arguments
        digits: size N x 64 numpy array with the images
        means: size 10 x 64 numpy array with the 10 class means
        covariances: size 10 x 64 x 64 numpy array with the 10 class covariances

    Returns
        likelihoods: size N x 10 numpy array with the ith row corresponding
              to logp(t | x^(i)) for t in {0, ..., 9}
    '''
    # == YOUR CODE GOES HERE ==
    N = digits.shape[0]
    likelihoods = np.zeros((N, 10))
    px = np.sum(np.exp(generative_likelihood(digits, means, covariances))
                * (1/10), axis=1)  # Find px using the law of total probability
    pk = 1/10
    px_t = generative_likelihood(digits, means, covariances)
    for t in range(10):
        p = px_t[:, t] - np.log(px) + np.log(pk)
        likelihoods[:, t] = p
    return likelihoods
    # ====
```

```python
def classify_data(digits, means, covariances):
    '''
    Classify new points by taking the most likely posterior class.
    Make sure that your code is vectorized.

    Arguments
        digits: size N x 64 numpy array with the images
        means: size 10 x 64 numpy array with the 10 class means
        covariances: size 10 x 64 x 64 numpy array with the 10 class covariances

    Returns
        pred: size N numpy array with the ith element corresponding
              to argmax_t log p(t | x^(i))
    '''
    # Compute and return the most likely class
    log = conditional_likelihood(digits, means, covariances)
    c = np.argmax(log, axis=1)  # Find the label that maximizes the likelihood
    return c
    # == YOUR CODE GOES HERE ==
    # ====
```

3b)

```
Train average conditional log-likelihood:  -0.12462443666863028
Test average conditional log-likelihood:  -0.19667320325525578
Train posterior accuracy:  0.9814285714285714
Test posterior accuracy:  0.97275
```