CSC 311 #3 : D Nov 15 2019

Now let's derive $L(\theta) = P(x_j, c \mid \theta, \pi)$

$$L(\theta) = P(x_j, c \mid \theta, \pi) = \pi_c \prod_{j=1}^{784} \theta_{cj}^{x_j} (1-\theta_{cj})^{(1-x_j)}$$

Question 1

1a)

① Derive for π_c

$$P(t \mid \pi) = \prod_{i=0}^{N} \prod_{j=0}^{9} \pi_j^{t_j^{(i)}}$$

$$\log(P(t \mid \pi)) = \log\left(\prod_{i=0}^{N} \prod_{j=0}^{9} \pi_j^{t_j^{(i)}}\right)$$

$$= \sum^{N} \sum^{9} \log(\pi_j^{t_j^{(i)}})$$

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

$$= \sum_{i=0}^{N}\sum_{j=0}^{8} \log\left(\pi_j^{t_j^{(i)}}\right) + \log\left(\left(1-\sum_{j=0}^{8}\pi_j\right)^{t_9^{(i)}}\right)$$ (by hint given)

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

Now, we take the partial derivative w.r.t π_j,

$$\frac{\partial A}{\partial \pi_j} = \sum_{i=0}^{N} \frac{t_j}{\pi_j} - \frac{t_9^{i}}{1-\sum_{j=0}^{8}\pi_j} = 0$$

$\Rightarrow$ since

$\pi_0 + \pi_1 + \pi_2 + \dots + \pi_9 = 1$

$\frac{\pi_1}{\pi_9} + \frac{\pi_2}{\pi_9} + \dots + 1 = \frac{1}{\pi_9}$

$$\frac{t_q^{(i)}}{1-\sum_{j=0}^{8}\pi_j} = \frac{t_j^{(i)}}{\pi_j}$$

$$\pi_q = \frac{1}{\sum_{i=0}^{8}\frac{\pi_i}{\pi_q}+1} = 1-\sum_{j=0}^{8}\pi_j \quad *$$

$$\hat{\pi}_j = \sum_{i=0}^{N} t_j^{(i)}\left(1-\sum_{j=0}^{8}\pi_j\right)\left(t_q^{(i)}\right)^{-1}$$

$$=$$

$$\pi_j = \sum_{i=0}^{N} t_j^{(i)} \left(\frac{1}{\sum_{j=0}^{8} \frac{\pi_j}{\pi_9} + 1} \right) (t_9^{(i)})^{-1}$$

Now back to $L(\theta)$

$$L(\theta) = P(x_j, c \mid \theta, \pi) = \pi_c \prod_{j=1}^{784} \theta_{cj}^{x_j} (1-\theta_{cj})^{(1-x_j)}$$

We will solve θ

$$\log(L(\theta)) = \log(\pi_c) + \log\left(\prod_{j=1}^{784} \theta_{cj}^{x_j} (1-\theta_{cj})^{(1-x_j)}\right)$$

Since we only care about terms with θ, we will ignore $\log(\pi_c)$ [we had solved before]

$$= \log\left(\prod_{i=1}^{N} \prod_{j=1}^{784} \theta_{cj}^{x_j^i} (1-\theta_{cj})^{(1-x_j^i)}\right)$$

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

$$= \sum_{i=1}^{N}\sum_{j=1}^{784} x_j^{(i)} \log(\theta_{cj}) + (1-x_j^{(i)})(\log(1-\theta_{cj}))$$

We will use $\mathbb{1}(C^i = c)$ to indicate that when $c^i = c$ then we will have matrix that has 0 every except for the row and col position that represent the class position will be 1.

$$\frac{\partial L}{\partial \theta_{cj}} = \sum_{i}^{N} \mathbb{1}(C^{i}=c)\left(\frac{x_j}{\theta_{cj}} - \frac{(1-x_j)}{(1-\theta_{cj})} \right)$$

$$0 = \sum_{i}^{N} \mathbb{1}(c^{i}=c) \frac{\left(x_j^{(i)}(1-\theta_{cj}) - (\theta_{cj})(1-x_j^{(i)}) \right)}{(\theta_{cj})(1-\theta_{cj})}$$

$$0 = \sum_{i}^{N} \mathbb{1}(c^{i}=c) \frac{\left(x_j^{(i)} - x_j^{(i)}\theta_{cj} - \theta_{cj} + x_j^{(i)}\theta_{cj} \right)}{\theta_{cj}(1-\theta_{cj})}$$

$$0 = \sum_{i}^{N} \mathbb{1}(c^{i}=c) \left(\frac{x_j^{(i)}}{\theta_{cj}(1-\theta_{cj})} - \frac{\theta_{cj}}{\theta_{cj}(1-\theta_{cj})} \right)$$

$$0 = \sum_i \mathbb{1}(c^i = c)\left(\frac{x_j^{(i)}}{\theta_{cj}(1-\theta_{cj})}\right) - \sum_i^N \mathbb{1}(c^i = c)\left(\frac{1}{1-\theta_{cj}}\right)$$

$$\sum_i^N \mathbb{1}(c^i = c)\left(\left(\frac{x_j^{(i)}}{\theta_{cj}}\right)\frac{1}{\cancel{(1-\theta_{cj})}}\right) = \sum_i^N \mathbb{1}(c^i = c)\left(1 \cdot \frac{1}{\cancel{1-\theta_{cj}}}\right)$$

$$\theta_{cj} = \frac{\sum_i^N \mathbb{1}(c^i = c)\, x_j^{(i)}}{\sum_i^N \mathbb{1}(c^i = c)}$$

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

Question 1 b)

$\log(p(t \mid x, \theta, \pi))$ for a single training image.

```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
    cl = np.transpose(train_images).dot(train_labels)
    cls = np.sum(train_labels, 0)
    theta_mle = cl/cls
    pi_mle = cls / (train_images.shape[0])
    return theta_mle, pi_mle


def train_map_estimator(train_images, train_labels):
    """ Inputs: train_images, train_labels
        Returns the MAP estimators theta_map and pi_map"""
    cl = np.transpose(train_images).dot(train_labels)
    cls = np.sum(train_labels, 0)
    theta_map = (cl + 2)/ (cls * 4)
    pi_map = cls / train_images.shape[0]
    return theta_map, pi_map
```

$$\log(P(C|X,\theta,\pi)) = \log P(X|C,\theta) + \log(P(C|\pi)) - \log P(X)$$

$$= \log \prod_{j=1}^{784} P(x_j|C,\theta_{jc}) + \log P(C|\pi) - \log \sum_{c=0}^{9} P(X|C=c)\,P(C=c)$$

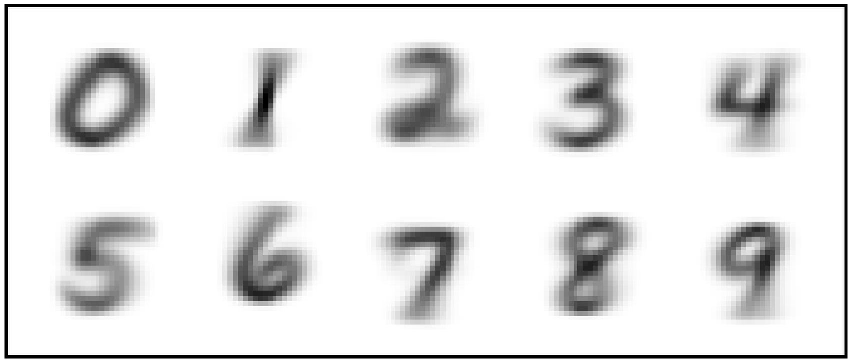
$$= \sum_{j=1}^{784} (x_j \log \theta_{cj} + (1-x_j)\log(1-\theta_{cj}) + \log \pi_c - \log \sum_{c=0}^{9} \pi_c \prod_{j=1}^{784} \theta_{cj}^{x_j}(1-\theta_{cj})^{(1-x_j)}$$

Question 1c)

When we try to report the average log likelihood per data a. nan value is returned due to the fact

1) we have a naive assumption on independence and hence less accurate compared to discriminative models.

Q1 a)



Q1 e)

$P(\theta \mid x, c, \pi) \propto P(\theta) P(x, c \mid \theta, \pi)$ where $\theta_{cj} \sim \text{Beta}(3,3)$

$L(\theta) = \theta_{cj}^{2} (1-\theta_{cj})^{2}$ $\quad P(x, c \mid \theta, \pi)$

$\log(L(\theta)) = 2\log(\theta_{cj}) + 2\log(1-\theta_{cj}) + \log(P(x, c \mid \theta, \pi))$

$= 2\log(\theta_{cj}) + 2\log(1-\theta_{cj}) +$

$$\sum_{j=1}^{784} x_j \log(\theta_{cj}) + (1-x_j)(\log(1-\theta_{cj}))$$

$+ C$ where C is the $\log(\pi_c)$.

Since we already derived the last part we will focus on the first part. ($2\log(\theta_{cj}) + 2\log(1-\theta_{cj})$)

$A = 2\log(\theta_{cj}) + 2\log(1-\theta_{cj})$

$$\frac{\partial A}{\partial \theta_{cj}} = \frac{2}{\theta_{cj}} - \frac{2}{1-\theta_{cj}} +$$

$$\sum_{i}^{N} \mathbb{1}(c^i=c)\left(\left(\frac{x_j}{\theta_{cj}}\right)\frac{1}{(1-\theta_{cj})}\right) - \sum_{i}^{N} \mathbb{1}(c^i=c)\left(1 \cdot \frac{1}{1-\theta_{cj}}\right)$$

(from Q1.a)

$$0 = \frac{2 - 2\cancel{\theta}c_j - 2\cancel{\theta}c_j}{\theta_{cj}(\cancel{1-\theta_{cj}})} + \sum_{i}^{N} \mathbb{1}(c^i=c)\left(\frac{x_j}{\theta_{cj}}\right)\left(\frac{1}{\cancel{1-\theta_{cj}}}\right) - \sum_{i}^{N} \mathbb{1}(c^i=c)\left(1 \cdot \frac{1}{\cancel{1-\theta_{cj}}}\right)$$

$$0 = \frac{-4\theta_{cj}+2}{\theta_{cj}} + \sum_{i}^{N} \mathbb{1}(c^i=c)\left(\frac{x_j}{\theta_{cj}}\right) - \sum_{i}^{N} \mathbb{1}(c^i=c)$$

$$0 = -4 + \frac{2}{\theta_{cj}} + \sum_{i}^{N} \mathbb{1}(c^i=c)\left(\frac{x_j}{\theta_{cj}}\right) - \sum_{i}^{N} \mathbb{1}(c^i=c)$$

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

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

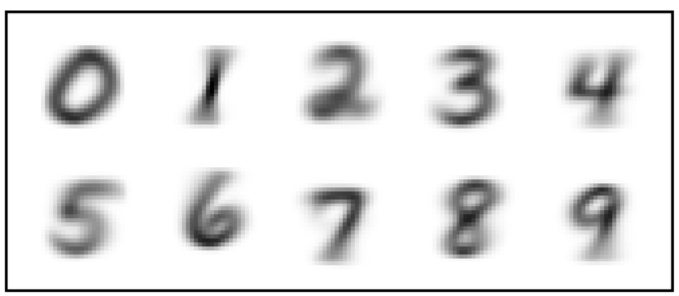
Q1f)

```
ef 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
    log_pi = np.log(pi)
    log_theta_images = images.dot(np.log(theta))
    log_inverse_theta = (1. - images).dot(np.log(1. - theta))
    log_like = log_pi + log_theta_images + log_inverse_theta
    x = logsumexp(log_like, axis=1)
    log_like = np.transpose(np.transpose(log_like)-x)
    return log_like
```

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

    return log_like == log_like.max(axis=1, keepdims=True)


ef 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
    predictions = predict(log_like)
    return np.mean(predictions == labels)
```

```
Average log-likelihood for MLE is  nan
Average log-likelihood for MAP is  -3.5323997704856405
Training accuracy for MAP is  0.9607233333333334
Test accuracy for MAP is  0.9574
```

Q1g)



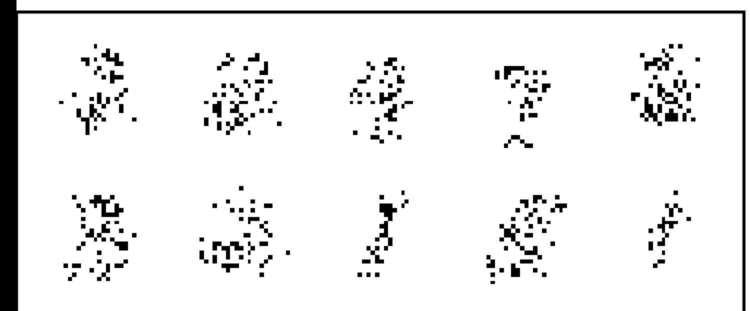
Q2) a) True

Q2.b) False

Q2.c)

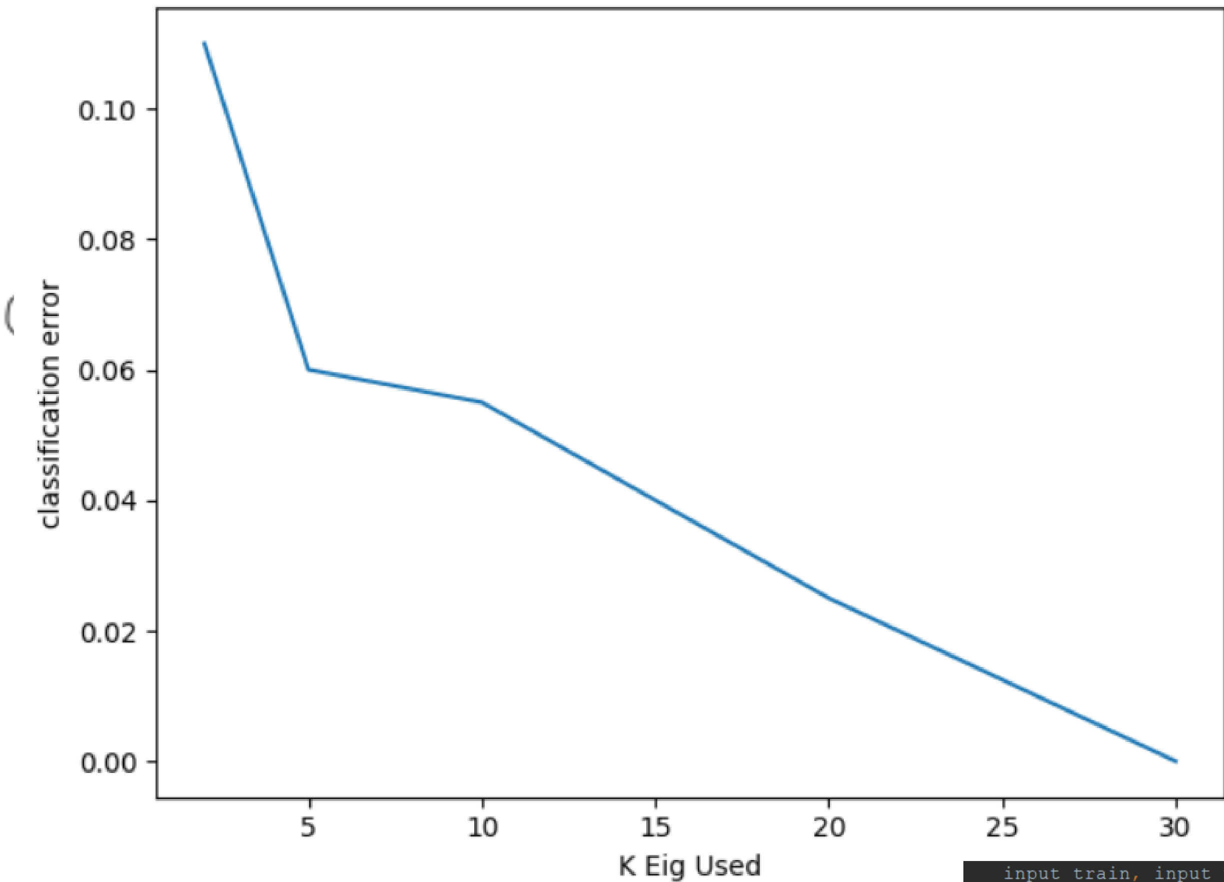
```python
def image_sampler(theta, pi, num_images):
    """ Inputs: parameters theta and pi, and number of images to sample
    Returns the sampled images"""

    # YOU NEED TO WRITE THIS PART
    sampled_images = np.zeros((num_images, 784))
    for i in range(num_images):
        c = np.random.choice(a=10, p=pi)
        print(c)
        sampled_images[i] = np.random.binomial(1, theta[:,c].reshape(1,784))
    return sampled_images
```

```
2
2
9
7
6
3
2
1
5
1
```



Q3 a)

Plot:




```
    input_train, input_valid, input_test, targ_train, targ_valid, targ_test = load_data(
"digits.npz")

    # PCA ALGORITHM
    # Subtract the mean from each dimension (centering)
    m = np.mean(input_train, axis=0)
    valid_centered = input_train - np.tile(m, (input_train.shape[0], 1))

    # Calculate the covariance matrix of the data;
    C = np.cov(valid_centered.T)
    # PCA (or equivalently SVD or EVD) SVD and EVD are equaivalent since C is symmetric PSD
    U, S, V = np.linalg.svd(C)
    # S is eigen
    # Project the data onto the first principal component, then back into 2D space
    kns = [2, 5, 10, 20, 30]
    class_err = []
    for k in kns:
        X_recon = valid_centered.dot(U[:, :k])
        nn_1 = run_knn( X_recon, targ_train, np.matmul(input_valid, U[:, :k]))
        acc = 0
        for p in range(len(nn_1)):
            if nn_1[p] == targ_valid[p]:
                acc += 1
        acc/= len(nn_1)
        print("acc is {} for k : {}".format(acc, k))

        class_err.append(1-acc)
```

(Q3b) I would choose K=30 since it

gave the best accuracy for validation testing which makes sense because we reduce our feature dimensions from 600 to 30 while still maintening the feature importance that influence the nearest neighbours classifier

Q3C) The performance on the final classifier over test data using

```
acc is 0.975 for k : 30
```