HOMEWORK 3
CSC311 Fall 2019
Nicole Xin Yue Wang, #1004235339

1. Fitting a Naïve Bayes Model

a) Let N be the number of inputs

so then dataset $X = [x^{(1)}, x^{(2)}, \dots, x^{(N)}]^T$

So we have:

$$\ell(\theta, \pi) = \sum_{i=1}^{N} \log p(x^{(i)}, t^{(i)} \mid \theta, \pi) = \sum_{i=1}^{N} \log \{ p(t^{(i)} \mid \pi) \prod_{j=1}^{784} p(x_j^{(i)} \mid t^{(i)}, \theta_j) \}$$

$$= \sum_{i=1}^{N} \Big[\underbrace{\log p(t^{(i)} \mid \pi)}_{\ell(\pi)} + \underbrace{\textstyle\sum_{j=1}^{784} \log p(x_j^{(i)} \mid t^{(i)}, \theta_j)}_{\ell(\theta)} \Big]$$

$$\ell(\theta_j) = \sum_{i=1}^{N} \log p(x_j^{(i)} \mid t^{(i)}, \theta_j) \} = \sum_{i=1}^{N} \sum_{c=0}^{9} t_c^{(i)} \left[x_j^{(i)} \log \theta_{jc} + (1 - x_j^{(i)}) \log (1 - \theta_{jc}) \right]$$

for pixel j and class c:

$$\frac{\partial \ell(\theta_j)}{\partial \theta_{jc}} = \sum_{i=1}^{N} t_c^{(i)} \left(\frac{x_j^{(i)}}{\theta_{jc}} - \frac{1 - x_j^{(i)}}{1 - \theta_{jc}} \right)$$

$$= \sum_{i=1}^{N} \frac{t_c^{(i)} (x_j^{(i)} - \theta_{jc})}{\theta_{jc}(1 - \theta_{jc})}$$

↳ set to zero: $\sum_{i=1}^{N} t_c^{(i)} x_j^{(i)} = \sum_{i=1}^{N} t_c^{(i)} \theta_{jc}$

$$\hat{\theta}_{jc} = \frac{\sum_{i=1}^{N} t_c^{(i)} x_j^{(i)}}{\sum_{i=1}^{N} t_c^{(i)}}$$

⇒ counts the number of images with a white pixel on j in all c labeled images.

$$\ell(\pi) = \sum_{i=1}^{N} \log p(t^{(i)} \mid \pi)$$

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

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

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

Let constraint be $\sum_{c=0}^{9} \pi_c = 1$, F be our Lagrange Multiplier

$$\Rightarrow F = \sum_{i=1}^{N} \Big[\sum_{j=0}^{8} t_j^{(i)} \log \pi_j + t_9^{(i)} \log (1 - \sum_{k=0}^{8} \pi_k) \Big] - \lambda \Big(\sum_{c=0}^{9} \pi_c - 1 \Big)$$

for $j = 0, \dots, 8$: $F_{\hat{\pi}_j} = \sum_{i=1}^{N} \left(\frac{t_j^{(i)}}{\hat{\pi}_j} - \frac{t_9^{(i)}}{\hat{\pi}_9} \right) - \lambda$

$$F_{\hat{\pi}_9} = \sum_{i=1}^{N} \frac{t_9^{(i)}}{\hat{\pi}_9} - \lambda$$

$$F_\lambda = -\sum_{c=0}^{9} \pi_c + 1$$

set $F_{\hat{\pi}_9} = 0 \Rightarrow \frac{1}{\hat{\pi}_9} \sum_{i=1}^{N} t_9^{(i)} = \lambda$

$$\hat{\pi}_9 = \frac{\sum_{i=1}^{N} t_9^{(i)}}{\lambda} \quad ①$$

set $F_{\hat{\pi}_j} = 0 \Rightarrow \frac{1}{\hat{\pi}_j} \sum t_j^{(i)} - \frac{1}{\hat{\pi}_9} \sum t_9^{(i)} = \lambda$

$$\Rightarrow \frac{1}{\hat{\pi}_j} \sum_{i=1}^{N} t_j^{(i)} - \frac{1}{\hat{\pi}_9} \sum_{i=1}^{N} t_9^{(i)} = \frac{1}{\hat{\pi}_9} \sum_{i=1}^{N} t_9^{(i)}$$

$$\frac{1}{\hat{\pi}_j} \sum_{i=1}^{N} t_j^{(i)} = \frac{2}{\hat{\pi}_9} \sum_{i=1}^{N} t_9^{(i)}$$

$$\frac{\hat{\pi}_j}{\hat{\pi}_9} = \frac{\sum_{i=1}^{N} t_j^{(i)}}{2 \sum_{i=1}^{N} t_9^{(i)}} \quad ②$$

$$F_\lambda = -\hat{\pi}_9 \sum_{c=0}^{9} \frac{\hat{\pi}_c}{\hat{\pi}_9} + 1$$

$$= \frac{-\sum_{i=1}^{N} t_9^{(i)}}{\lambda} \left[\sum_{c=0}^{8} \frac{\sum_{i=1}^{N} t_c^{(i)}}{2 \sum_{i=1}^{N} t_9^{(i)}} + 1 \right] + 1$$

$$= -\frac{1}{2\lambda} \sum_{i=1}^{N} \sum_{c=0}^{8} t_c^{(i)} - \frac{\sum_{i=1}^{N} t_9^{(i)}}{\lambda} + 1$$

#1. a) continue...

set $F_\lambda = 0$

$$\Rightarrow \quad 0 = \sum_{i=1}^{N} \sum_{c=}^{8} t_c^{(i)} + 2\sum_{i=1}^{N} t_9^{(i)} - 2\lambda$$

$$0 = N + \sum_{i=1}^{N} t_9^{(i)} - 2\lambda \qquad \# \sum_{c=0}^{9} t_c^{(i)} = 1$$

$$\lambda = \frac{1}{2}\left(N + \sum_{i=1}^{N} t_9^{(i)}\right)$$

plug λ into ①: $\quad \hat{\pi}_9 = \dfrac{2\sum_{i=1}^{N} t_9^{(i)}}{N + \sum_{i=1}^{N} t_9^{(i)}}$

plug $\hat{\pi}_9$ into ②: $\quad \hat{\pi}_j = \dfrac{2\sum_{i=1}^{N} t_9^{(i)}}{N + \sum_{i=1}^{N} t_9^{(i)}} \cdot \dfrac{\sum_{i=1}^{N} t_j^{(i)}}{2\sum_{i=1}^{N} t_9^{(i)}}$

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

b) By Bayes' Rule:

$$p(t^{(i)} | x^{(i)}, \theta, \pi) = \frac{p(x^{(i)}, t^{(i)} | \theta, \pi)}{p(x|\theta,\pi)}$$

$$= \frac{p(t^{(i)}|\pi) \prod_{j=1}^{784} p(x_j^{(i)} | t^{(i)}, \theta_{jt^{(i)}})}{\sum_{c=0}^{9} p(x^{(i)} | c, \theta, \pi) p(c|\theta,\pi)}$$

$$= \frac{\pi_{t^{(i)}} \prod_{j=1}^{784} p(x_j^{(i)} | t^{(i)}, \theta_{jt^{(i)}})}{\sum_{c=0}^{9} \pi_c \prod_{j=1}^{784} p(x_j^{(i)} | c, \theta_{jc})}$$

$$= \frac{\pi_{t^{(i)}} \prod_{j=1}^{784} \theta_{jt^{(i)}}^{x_j^{(i)}} (1-\theta_{jt^{(i)}})^{1-x_j^{(i)}}}{\sum_{c=0}^{9} \left[\pi_c \prod_{j=1}^{784} \theta_{jc}^{x_j^{(i)}} (1-\theta_{jc})^{1-x_j^{(i)}}\right]}$$

$$\therefore \log p(t^{(i)} | x^{(i)}, \theta, \pi) = \log \pi_{t^{(i)}} + \sum_{j=1}^{784}\left[x_j^{(i)} \log \theta_{jt^{(i)}} + (1-x_j^{(i)}) \log(1-\theta_{jt^{(i)}})\right] - \log \sum_{c=0}^{9}\left[\pi_c \prod_{j=1}^{784} \theta_{jc}^{x_j^{(i)}} (1-\theta_{jc})^{1-x_j^{(i)}}\right]$$

Code for log_likelihood function:

```
def log_likelihood(images, theta, pi):
image_num = np.shape(images)[0]
    class_num = np.shape(theta)[1]

    log_like = np.zeros((image_num, class_num))
    for i in range(image_num):
        for c in range(class_num):
            nume = pi[c] * np.prod(np.power(theta[:,c], images[i]) *
                         np.power(1-theta[:,c], 1-images[i]))
            if(nume != 0):
                denom = 0
                for cc in range(class_num):
                    denom += pi[cc] * np.prod(np.power(theta[:,cc], images[i]) *
                                  np.power(1-theta[:,cc], 1-images[i]))
                log_like[i][c] = np.log(nume / denom)
            else:
                log_like[i][c] = 0

    return log_like
```

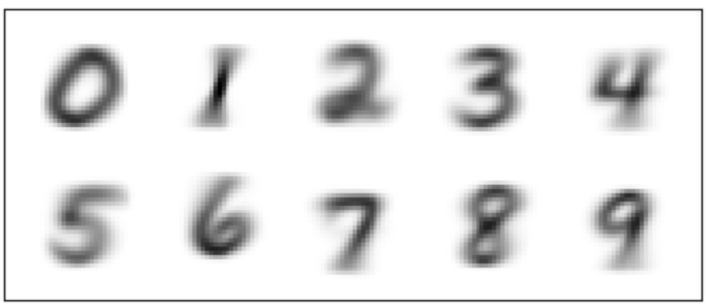
c) fitting θ and π using MLE

```
def train_mle_estimator(train_images, train_labels):
    (image_num, pixel_num) = np.shape(train_images)
    class_num = np.shape(train_labels)[1]

    theta_mle = np.zeros((pixel_num, class_num))
    pi_mle = np.zeros(class_num)

    for c in range(class_num):
        tc = train_labels[:, c].copy()
        for j in range(pixel_num):
            xj = train_images[:, j].copy()
            theta_mle[j,c] = np.sum(tc * xj) / np.sum(tc)
        if(c == 9):
            pi_mle[c] = 2 * np.sum(tc) / (image_num + np.sum(train_labels[:, 9]))
        else:
            pi_mle[c] = np.sum(tc) / (image_num + np.sum(train_labels[:, 9]))

    return theta_mle, pi_mle
```

Problem: there exist cases where θ_{jc} is 0, then it will be unable to find the log-likelihood since log of zero is N/A. If I set log-likelihood for where θ_{jc} is 0, I will get:

Average log-likelihood for MLE is -3.314481563095963

d) 10 greyscale images of MLE estimator plotted, one for each class



e)

$$\log p(\theta, x|\pi) = \log[p(\theta)\, p(x^{(i)}|\theta,\pi)] = \log p(\theta) + \ell_{MLE}$$

Using Beta(3,3) as prior:

$$p(\theta, 3, 3) \propto \theta^2 (1-\theta)^2$$

$$\Rightarrow \log p(\theta) = 2\log\theta + 2\log(1-\theta)$$

$$\log p(\theta, x|\pi) = 2\log\theta + 2\log(1-\theta) + \ell_{MLE}$$

$$\frac{\partial}{\partial \theta_{jc}} \log p(\theta, x|\pi) = \frac{2}{\theta_{jc}} - \frac{2}{1-\theta_{jc}} + \sum_{i=1}^{N} \frac{t_c^{(i)}(x_j^{(i)} - \theta_{jc})}{\theta_{jc}(1-\theta_{jc})}$$

$$= \frac{2 - 4\theta_{jc}}{\theta_{jc}(1-\theta_{jc})} + \sum_{i=1}^{N} \frac{t_c^{(i)}(x_j^{(i)} - \theta_{jc})}{\theta_{jc}(1-\theta_{jc})}$$

$$= \frac{2 - 4\theta_{jc} + \sum_{i=1}^{N} t_c^{(i)}(x_j^{(i)} - \theta_{jc})}{\theta_{jc}(1-\theta_{jc})}$$

↳ set to zero $\Rightarrow 2 - 4\theta_{jc} + \sum_{i=1}^{N} t_c^{(i)} x_j^{(i)} - \theta_{jc}\sum_{i=1}^{N} t_c^{(i)} = 0$

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

f) fitting θ and π using MAP and finding accuracy.

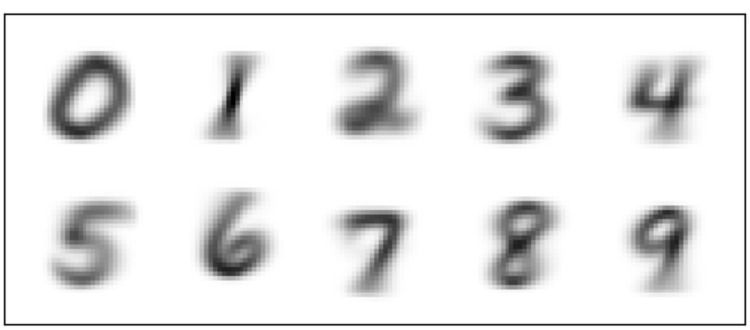
```
def train_map_estimator(train_images, train_labels):
    (image_num, pixel_num) = np.shape(train_images)
    class_num = np.shape(train_labels)[1]
    theta_map = np.zeros((pixel_num, class_num))
    pi_map = np.ones(class_num)

    for c in range(class_num):
        tc = train_labels[:, c].copy()
        for j in range(pixel_num):
            xj = train_images[:, j].copy()
            theta_jc = (2 + np.sum(tc * xj)) / (4 + np.sum(tc))
            theta_map[j][c] = theta_jc
        if(c == 9):
            pi_map[c] = 2 * np.sum(tc) / (image_num + np.sum(train_labels[:, 9]))
        else:
            pi_map[c] = np.sum(tc) / (image_num + np.sum(train_labels[:, 9]))
    return theta_map, pi_map


def predict(log_like):
    image_num = np.shape(log_like)[0]
    predictions = np.zeros(np.shape(log_like))
    for i in range(image_num):
        max_index = np.argmax(log_like[i])
        predictions[i][max_index] = 1
    return predictions

def accuracy(log_like, labels):
    predictions = predict(log_like)
    match_num = 0
    N = np.shape(labels)[0]
    for i in range(N):
        if(np.array_equal(predictions[i], labels[i])):
            match_num += 1
    accuracy = match_num / N
    return accuracy
```

Average log-likelihood for MAP is -3.3669909342818136

Training accuracy for MAP is 0.83085

Test accuracy for MAP is 0.809

g) 10 greyscale images of MAP estimator plotted, one for each class



2. Generating from a Naïve Bayes Model

a) True

b) False

c) Producing random images

```
def image_sampler(theta, pi, num_images):
    (pixel_num, class_num) = np.shape(theta)
    sampled_images = np.zeros((num_images, pixel_num))
    cs = np.random.choice(class_num, num_images, p=pi)
    for i in range(num_images):
        sampled_images[i] = np.random.binomial(1, p=theta[:, cs[i]])
    print("The random classes are: " + str(cs))
    return sampled_images
```

Output:

The random classes are: [7 0 7 9 8 3 6 8 2 8]