Exercise 1

i) Lagrangian: $g(\lambda, w) = L(w) - \lambda \underbrace{\left(\sum_{i=1}^{M} w_i - 1\right)}_{g(w)}$

Now we want to solve for the langaian to find the argmax $L(w)$ given the constraints, we can do so by solving the following:

$$\nabla g(\lambda, w) = 0 \iff \begin{cases} \nabla L(w) - \lambda \nabla g(w) = 0 \\ g(w) = 0 \end{cases} \quad \text{(I)}$$

so now:

$$g(\lambda, w) = \sum_{i=1}^{M} c_i \log w_i - \lambda\left(\sum_{i=1}^{M} w_i - 1\right)$$

taking the derivative in terms of w:

$$\frac{\partial g(\lambda, w)}{\partial w_a} = \sum_{i=1}^{M} \frac{c}{w_{ai}} - \lambda \sum_{i=1}^{M} 1 \qquad c w^{-}$$

$$= \left[\frac{c_1}{w_1}, \ldots, \frac{c_M}{w_M}\right]^T - \lambda [1, \ldots, 1]^T = 0$$

Now rearranging this gives us:

$$\left[\frac{c_1}{w_1}, \ldots, \frac{c_M}{w_M}\right]^T = \lambda [1, \ldots, 1]^T$$

$$\nabla L(w) - \lambda \nabla g(w) = 0$$

where $\lambda = \left[\frac{c_1}{w_1}, \ldots, \frac{c_M}{w_M}\right]^T$ and $w \in \bar{\mathcal{F}}$ giving us extrema value which satisfies our constraint and thus satisfies the first part of (I), and furthermore we need $g(w) = \sum_{i=1}^{M} w_i - 1 = 0$, so then we can see the constraint set $\bar{\mathcal{F}} = \{w \in \mathbb{R}^M \mid \sum_{i=1}^{M} w_i = 1\}$ is satisfied giving us the desired result. Now

$$\nabla L(w) = \left[\frac{c_1}{w_1}, \ldots, \frac{c_M}{w_M}\right]^T = 0$$

states that $w \in \mathcal{F}$ is an extrema, lets use the second derivative test to show that w is a maximum

$$\nabla_{ww} L(w) = \sum_{i=1}^{M} \frac{c_{iab}}{w_{iab}}$$

(II) $$= \sum_{i=1}^{M} -\frac{(c_{ab})_i}{(w_{ab})_i^2}$$

← we know that $c_1, \ldots, c_M \geq 0$

← denominator will always be positive since it is squared

And we can see that $\nabla_{ww} L(w) < 0$, by the second derivative test since our second derivative is negative at our only extrema $w \in \mathcal{F}$, we see that our extrema is a global maximum over $\mathcal{F}$.

ii) Note that in part i), we did not explicitly mention that $w_i > 0$, but it must be assumed implicitly as $\log(w_i)$ must take in positive w_i values as logarithms cannot have non-positive arguments. So adding this fact to part i) allows us to explicity state that

$$\hat{w} = \left[\frac{c_1}{\sum_{i=1}^{M} c_i}, \ldots, \frac{c_2}{\sum_{i=1}^{M} c_i}\right]$$

is a global maximum, as (II) shows us that the second derivative will give us a negative value where $w \in \mathcal{B}$

Excercise 2

i) we have that $N = 14$

Now let: $\hat{q}_0 = 0 =$ 'does not buy computer'

$\hat{q}_1 = 1 =$ 'buy computer'

where $c_0 := \sum_{s=1}^{14} \mathbb{1}(y_s = '0') = 4$

$c_1 := \sum_{s=1}^{14} \mathbb{1}(y_s = '1') = 10$

now normalizing the above quantity gives us a prior PMF of:

$$[\hat{q}_0, \hat{q}_1] = [0.29, 0.71]$$

$\hat{q}_0$: does not buy computer

$\hat{q}_1$: buys computer

ii)

class conditional PMF for not buying computer:

age: ≤30, [31,40], >40 | income: low, med, high | student: no, yes | credit rating: fair, excel

$$[\hat{q}_{0,1}, \ldots, \hat{q}_{0,10}] = \frac{1}{16}[2, 0, 2, 1, 1, 2, 3, 1, 1, 3]$$

class conditional PMF for buying computer:

age: ≤30, [31,40], >40 | income: low, med, high | student: no, yes | credit rating: fair, excel

$$[\hat{q}_{1,1}, \ldots, \hat{q}_{1,10}] = \frac{1}{40}[3, 4, 3, 3, 5, 2, 5, 5, 7, 3]$$

iii) Compute the predictive probabilites for the following:

x_1 = "age ≤ 30, medium income, student, fair credit rating"

age: ≤30 [31,40] >40 | income: low med high | student: no yes | credit rating: fair excel

$$\phi(x_1) = [1, 0, 0, 0, 1, 0, 0, 1, 1, 0]$$

$$P(y=0 \mid \phi(x_1)) \propto \left[\frac{2^1 \cdot 0^0 \cdot 2^0 \cdot 1^0 \cdot 1^1 \cdot 2^0 \cdot 3^0 \cdot 1^1 \cdot 1^1 \cdot 3^0}{16^4}(0.29)\right] = 7.93\ E^{-6}$$

$$P(y=1 \mid \phi(x_1)) \propto \left[\frac{3^1 \cdot 4^0 \cdot 3^0 \cdot 3^0 \cdot 5^1 \cdot 2^0 \cdot 5^0 \cdot 5^1 \cdot 7^1 \cdot 3^0}{40^4}(0.71)\right] = 0.00014$$

Now normalizing gives us:

$$P(y=0 \mid \phi(x_1)) = \frac{7.93E^{-6}}{0.00014793} = 0.05$$

$$P(y=1 \mid \phi(x_1)) = \frac{0.00014}{0.00014793} = 0.95$$

I predict that test sample x_1 will choose to buy a computer as our model gave us a 95% chance that this will happen. The factor that most likely contributed to this is the fair credit rating.

x_2 = "age ∈ [31,40], low income, not student, fair credit rating"

age: ≤30 [31,40] >40 | income: low med high | student: no yes | credit rating: fair excel

$$\phi(x_2) = [0, 1, 0, 1, 0, 0, 1, 0, 1, 0]$$

$$P(y=0 \mid \phi(x_2)) \propto \left[\frac{2^1 \cdot 0^0 \cdot 2^0 \cdot 1^1 \cdot 1^0 \cdot 2^0 \cdot 3^1 \cdot 1^0 \cdot 1^1 \cdot 3^0}{16^4}(0.29)\right] = 0.00001$$

$$P(y=1 \mid \phi(x_2)) \propto \left[\frac{3^0 \cdot 4^1 \cdot 3^0 \cdot 3^1 \cdot 5^0 \cdot 2^0 \cdot 5^1 \cdot 5^1 \cdot 7^1 \cdot 3^0}{40^4}(0.71)\right] = 0.00011$$

$$P(y=0 \mid \phi(x_2)) = \frac{0.00001}{0.00012} = 0.08$$

$$P(y=1 \mid \phi(x_2)) = \frac{0.00011}{0.00012} = 0.92$$

I predict x_2 will end up buying the computer. Once again the factor that affected this prediction the most is the 'fair credit rating'.

age: ≤30 [31,40] >40 | income: low med high | student: no yes | credit rating: fair excel

$$\phi(x_3) = [0, 0, 1, 0, 0, 1, 1, 0, 0, 1]$$

$$P(y=0 \mid \phi(x_3)) \propto \left[\frac{2^1 \cdot 0^0 \cdot 2^1 \cdot 1^0 \cdot 1^0 \cdot 2^1 \cdot 3^1 \cdot 1^0 \cdot 1^0 \cdot 3^1}{16^4}(0.29)\right] = 0.00015$$

$$P(y=1 \mid \phi(x_3)) \propto \left[\frac{3^0 \cdot 4^1 \cdot 3^1 \cdot 3^0 \cdot 5^0 \cdot 2^1 \cdot 5^1 \cdot 5^0 \cdot 7^0 \cdot 3^1}{40^4}(0.71)\right] = 0.00002$$

$$P(y=0 \mid \phi(x_3)) = \frac{0.00015}{0.00017} = 0.88$$

$$P(y=1 \mid \phi(x_3)) = \frac{0.00002}{0.00017} = 0.12$$

I predict that x_3 will NOT buy a computer, as our model gave a 88% of this happening. The attributes that contributed to this the most is the fact that they are NOT students and their excellent credit rating.

iv) I went ahead and used the dataset given to us by the professor. To process the data I used the countvectorizer that returns a matrix w/ words as rows and docs as columns.

```
#vectorizer = TfidfVectorizer(stop_words=stopwords_list)
vectorizer = CountVectorizer(stop_words=stopwords_list)
vectors = vectorizer.fit_transform(data_cleaned).transpose()  # words x docs  # in the form of sparse matrix
idx_to_word = np.array(vectorizer.get_feature_names())   # list of words that corresponds to feature coordinates

print('>>>> vectors.shape', vectors.shape)
i = 3
print('newsgroups_labels[i] (catergory)', newsgroups_labels[i])
#print('>>>> data_cleaned[i] (article txt)', data_cleaned[i])
print('>>>> vectors[:,i] \n', vectors[:,i])
offset = 5545
C0 = 0
C1 = 0
q0 = [0] * 45534
q1 = [0] * 45534
q0_total = 0
q1_total = 0
y_test = [0] * 20
q_test = [0] * 45534
C_test = 0

#gets our train data to compute
for i in range(len(newsgroups_labels)-offset): #range ensures we get about 120 train cases
    if newsgroups_labels[i] == 0: #only looking for entries with label 1 or 0
        C0 += 1
        for j in range(45534):
            q0[j] += vectors[j,i]
            q0_total += vectors[j,i]
    elif newsgroups_labels[i] == 1: #see above ^
        C1 += 1
        for j in range(45534):
            q1[j] += vectors[j, i]
            q1_total += vectors[j,i]

q0_prior_pmf = (C0/(C0+C1))
q1_prior_pmf = (C1/(C0+C1))
q0_inflated = [x+1 for x in q0] #Added one to all entires in order to avoid multiplication by 0 in denominator
q1_inflated = [x+1 for x in q1]
#print('q0_total: ', q0_total)
#print('q1_total: ', q1_total)
#gets our test data
test_size = 0
for i in range(len(newsgroups_labels)-offset, len(newsgroups_labels)-(offset-40)): # -- range ensures we get about 30 test cases
    if newsgroups_labels[i] == 0 or newsgroups_labels[i] == 1:
        print('True label: ', newsgroups_labels[i], end=' ')
        C_test += 1
        y_word_count = 0
        numerator_0 = 1
        numerator_1 = 1
        for j in range(45534): # -- goes through the row of words, computes numerator
            if(vectors[j,i] != 0):
                y_word_count += vectors[j,i]
                exp = vectors[j,i]
                #if(q0[j] != 0 and q1[j] != 0):
                    #numerator_0 *= (q0[j] ** exp)
                    #numerator_1 *= (q1[j] ** exp)
                numerator_0 *= (q0_inflated[j] ** exp)
                numerator_1 *= (q1_inflated[j] ** exp)
                #print(' This is final numerator 0: ',numerator_0)
                #print(' This is final numerator 1: ',numerator_1)
        predict_prob_0 = ((numerator_0*q0_prior_pmf)/(q0_total ** y_word_count)) # -- computed predicted probabilites
        predict_prob_1 = ((numerator_1*q1_prior_pmf)/(q1_total ** y_word_count))
        pp0_norm = (predict_prob_0/(predict_prob_0 + predict_prob_1))
        pp1_norm = (predict_prob_1/(predict_prob_0 + predict_prob_1))
        print('Predicted probability of 0: ', pp0_norm, end=' ')
        print('Predicted probability of 1: ', pp1_norm)
```

I did although experience a handful amount of overflow issues in the final computation which would sometimes return negative probabilities, but for the most cases of our test cases the model did a relatively well job predicting the newspaper articles.