Naive Bayes and EM Algorithm

By Zach Wood-Doughty;
adapted from a similar writeup by Michael Collins

November 9, 2022: v1

1 Introduction and notation

This note will recap the Naive Bayes method and derive the maximum-likelihood estimate in the fully-supervised setting. Then we will introduce the expectation-maximization (EM) algorithm for the case where (some) labels are missing from the training data.

Suppose we want to use a dataset of email messages for the supervised learning task of classifying whether an email is spam. We'll represent our dataset of N examples as $\{X_i, y_i\}_{i=1}^N$. X_i is a vector of shape (1, V), where $X_{i,j}$ is the number of times the jth word appears in email i, and V is the number of words in our vocabulary. This is sometimes referred to as a "bag-of-words" representation. y_i is the label for the corresponding email, which we will assume is either 1 (if spam) or 0 (if not). We'll assume there exists some (noisy) target function $f : X \to y$ such that $y_i = f(X_i)$, and our goal is to learn a hypothesis h such that $h(X) \approx f(X)$. We'll denote the parameters of our model as Θ, such that our hypothesis $h(X)$ is uniquely determined by Θ. Our goal is to learn a Θ from the data such that $h(X)$ is as close as possible to $f(X)$.

2 The Naive Bayes model

Naive Bayes is a *generative* probabilistic model, meaning that it can model the entire distribution $p(X, y)$. This means we can use it both to predict the label of a test set example and sample new examples that should 'look similar' to the existing examples. To make it easier to distinguish two kinds of parameters in our model, we will write $\Theta = \{\alpha, \beta\}$. The α parameters define the overall log probability of the labels, with $\log p(y_i = y) = \alpha_y$. The β parameters define the log probability of seeing the jth word (once) in a document with label y as $\log p(w_j \mid y_i = y) = \beta_{j,y}$

In our example of binary spam classification, $p(y_i)$ is the baseline probability that an email is spam, *without* looking at the text of the email. So if on average only 1% of emails are spam, we would want to learn $\alpha = [-0.01, -4.61]$ such that $\log p(y_i = 1) = \alpha_1$ for any i, and so $p(y_i = 1) = \exp(\alpha_1) = \exp(-4.61) = 0.01$.[1]

$\log p(X_{i,j} \mid y_i; \beta)$ is the log probability of jth word being used $X_{i,j}$ times in an email with label y_i. Because in this assignment we'll assume that y is binary, β is a matrix of shape $[V, 2]$ (where

[1]Note that it's possible to represent α as a single parameter rather than a vector of two parameters that sum to one. The homework test cases will expect you to represent this as a vector of two parameters; this will also make easier to use numpy's matrix multiplication operations.

V is again the size of the vocabulary).[2] For example, if we think that "jackpot" comprises 0.3% of the words used in spam emails but only 0.06% of words in non-spam emails, then the we would have the row of β corresponding to the word "jackpot" would look like $[-5.81, -7.42]$.

Our goal is to find the values of α and β that *best explain* the data that we have. Remember from lecture that we wrote the derivation for finding the maximum-likelihood Θ^* as:

$$\begin{aligned}
\Theta^* &= \arg\max_{\Theta} p(\Theta \mid X, y) \\
&= \arg\max_{\Theta} \frac{p(X, y \mid \Theta)p(\Theta)}{p(X, y)} \\
&= \arg\max_{\Theta} p(X, y \mid \Theta)p(\Theta) \qquad (1) \\
&= \arg\max_{\Theta} p(X, y \mid \Theta) \qquad (2)
\end{aligned}$$

In (2), the $p(X, y \mid \Theta)$ term is referred the *likelihood* of the data, or the probability of seeing this particular dataset given our current parameters for the model.[3] While we will want to use our model to *predict* Y_i from X_i using the conditional distribution $p(Y_i \mid X_i)$, our model parameters α and β instead represent $\log p(Y_i)$ and $\log p(X_i \mid Y_i)$.

Because X_i is a vector with V values, writing out the entire conditional distribution would require $\mathcal{O}(2^V)$ parameters. For a reasonably large vocabulary of thousands of words, this is impossible. Instead, we will drastically simplify the problem by assuming that the occurrences of the jth word are *independent* of occurrences of all other words, conditional on the label y. This is an unrealistic (*naive*) assumption; for example, we might expect the word "win" to co-occur with the word "jackpot" in both spam and non-spam emails. The Naive Bayes assumption says no: $p(w_j, w_k \mid y) = p(w_j \mid y)p(w_k \mid y)$ for all words w_j and w_k. In plain language, we're assuming that how many times "jackpot" appears depends *only* on whether the email is spam or not; it does not depend on any of the other words in the email. This assumption allows us to write:

$$p(X_{i,1}, X_{i,2}, \ldots X_{i,V} \mid Y_i) = \prod_{j=1}^{V} p(X_{i,j} \mid Y_i) \qquad (3)$$

This assumption also encodes independence between any two occurrences of the same word. That is, the probability of the word "jackpot" appearing twice is equal to the probability of it appearing once, squared. More generally, each entry $X_{i,j}$ in our data matrix indicates the number of times the jth word appears in the ith document. To compute the log probability of the jth word appearing $X_{i,j}$ times, we write $\log p(X_{i,j} \mid y_i = y) = X_{i,j} \log p(w_j \mid y_i = y) = X_{i,j}\beta_{j,y}$. Here, we're using the fact that $\log(a^b) = b \log(a)$.

[2]If y were not binary and instead had K labels, then α would be an array of K values that sum to 1, and β would be a matrix of shape $[V, K]$ where each *column* sums to 1 but each row does not.

[3]In this derivation, (1) is the maximum a posteriori (MAP) estimate for Θ^*. By dropping the prior (the $p(\Theta)$ term), we get the maximum likelihood estimate, which assumes that all Θ) values are equally likely. For this assignment, we will only consider maximum likelihood, not MAP.

We can put this all together to connect $\Theta^* = \{\alpha^*, \beta^*\}$ for our dataset of N examples as follows:

$$
\begin{aligned}
\Theta^* &= \arg\max_{\Theta} \prod_{i=1}^{N} p(X_i, y_i \mid \Theta) \\
&= \arg\max_{\Theta} \log \prod_{i=1}^{N} p(X_i, y_i \mid \Theta) \qquad (4) \\
&= \arg\max_{\Theta} \sum_{i=1}^{N} \log p(X_i, y_i \mid \Theta) \\
&= \arg\max_{\Theta} \sum_{i=1}^{N} \log p(y_i \mid \alpha) + \sum_{i=1}^{N} \log p(X_i \mid y_i, \beta) \\
&= \arg\max_{\Theta} \sum_{i=1}^{N} \log p(y_i \mid \alpha) + \sum_{i=1}^{N} \sum_{j=1}^{V} \log p(X_{i,j} \mid y_i, \beta) \\
&= \arg\max_{\Theta} \sum_{i=1}^{N} \alpha_{y_i} + \sum_{i=1}^{N} \sum_{j=1}^{V} X_{i,j} \beta_{j,y_i} \qquad (5) \\
&= \arg\max_{\Theta} \mathcal{L}(X, y, \alpha, \beta)
\end{aligned}
$$

We'll use $\mathcal{L}(X, y, \alpha, \beta)$ to denote the log likelihood of the data and our specific parameters α and β. Fitting our model will be the process of finding $\arg\max_{\alpha,\beta} \mathcal{L}$. Note that (4) holds because log is monotonic (i.e., $\arg\max_x f(x) = \arg\max_x \log f(x)$).

Once we have learned α^* and β^*, we'll want to be able to use it to predict if an email is spam. This can be written as $p(y_i \mid X_i)$, or "what is the probability that the label y_i is 1 or 0 given that the email's text is X_i?" You might have noticed that we our parameters α and β aren't defined in terms of the probability distribution $p(y_i \mid X_i)$, but rather $p(y_i)$ and $p(X_{i,j} \mid y_i)$. This is because it's easier to track parameters in this way, and we can write our predictive probability as:

$$
p(y_i \mid X_i) = \frac{p(y_i|\alpha) p(X_i \mid y_i, \beta)}{p(X_i)} = \frac{p(y_i|\alpha) p(X_i \mid y_i, \beta)}{\sum_{y'=1}^{2} p(y'|\alpha) p(X_i \mid y', \beta)} \qquad (6)
$$

$$
\begin{aligned}
\log p(y_i \mid X_i) &= \log p(y_i|\alpha) + \log p(X_i \mid y_i, \beta) - \log \left(\sum_{y'=1}^{2} p(X_i \mid y', \beta) p(y'|\alpha) \right) \qquad (7) \\
&\propto \log p(y_i|\alpha) + \log p(X_i \mid y_i, \beta) \qquad (8) \\
&= \log p(y_i|\alpha) + \sum_{j=1}^{V} \log p(X_{i,j} \mid y_i, \beta) \\
&= \alpha_{y_i} + \sum_{j=1}^{V} X_{i,j} \beta_{j,y_i} \qquad (9)
\end{aligned}
$$

Before equation (8), the $\propto$ symbol means "is proportional to"; this is true because the summation inside the log term in (7) is the same for all X_i, regardless of the label being predicted. In your implementation, you can calculate $\log p(y_i \mid X_i) \propto \alpha_{y_i} + \sum_{j=1}^{V} X_{i,j} \beta_{j,y_i}$ for both possible values of y_i, and then use your `softmax` function to turn these unnormalized log probabilities into probabilities between 0 and 1.

3 Fully-supervised Naive Bayes updates

Remember that our dataset looks like $\{X_i, y_i\}_{i=1}^N$ and we've thus far assumed that every X_i has a corresponding y_i. In this setting, how do we learn good values for α and β? It turns out it is a reasonably simple process of counting:

$$\alpha_y = \log\left(\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(y_i = y)\right) \tag{10}$$

$$\beta_{j,y} = \log\left(\frac{\sum_{i=1}^{N} X_{i,j}\mathbb{1}(y_i = y)}{\sum_{i=1}^{N}\sum_{j=1}^{V} X_{i,j}\mathbb{1}(y_i = y)}\right) \tag{11}$$

Note that $\mathbb{1}(Z)$ is an *indicator* function which takes the value 1 if Z is true and 0 if Z is false. A derivation of these updates is shown in Appendix A.

4 Fully-unsupervised Naive Bayes

Suppose we want to use these same definitions of α and β even though we are now without labels. Instead of y being the observed labels, we will treat them as unobserved (latent) variables that define clusters which separate the data into groups. How does this change our derivation from (5)? For each example in our dataset, we need to consider the probability it has label y', for all possible values of y'! We can write our *unsupervised* log likelihood as:

$$\begin{aligned}
\mathcal{L}(X, \alpha, \beta) &= \log\prod_{i=1}^{N} p(X_i \mid \alpha, \beta) \\
&= \log\prod_{i=1}^{N}\sum_{y'=1}^{2} p(X_i, y_i = y' \mid \alpha, \beta) \\
&= \log\prod_{i=1}^{N}\sum_{y'=1}^{2} p(y_i = y' \mid X_i, \alpha, \beta)p(y' \mid \alpha)\prod_{j=1}^{V} p(X_{i,j} \mid y', \beta) && (12) \\
&= \sum_{i=1}^{N}\log\sum_{y'=1}^{2} p(y_i = y' \mid X_i, \alpha, \beta)p(y' \mid \alpha)\prod_{j=1}^{V} p(X_{i,j} \mid y', \beta) \\
&= \sum_{i=1}^{N}\log\sum_{y'=1}^{2} \exp\log\left(p(y_i = y' \mid X_i, \alpha, \beta)p(y' \mid \alpha)\prod_{j=1}^{V} p(X_{i,j} \mid y', \beta)\right) && (13) \\
&= \sum_{i=1}^{N}\log\sum_{y'=1}^{2} \exp\left(\log p(y_i = y' \mid X_i, \alpha, \beta) + \log p(y' \mid \alpha) + \sum_{j=1}^{V}\log p(X_{i,j} \mid y', \beta)\right) \\
&= \sum_{i=1}^{N}\log\sum_{y'=1}^{2} \exp\left(\log p(y_i = y' \mid X_i, \alpha, \beta) + \alpha_{y'} + \sum_{j=1}^{V} X_{i,j}\beta_{j,y'}\right) && (14)
\end{aligned}$$

If you compare (14) against (5), you should see many similarities. However, there are important differences.

One difference is that we cannot distribute the log inside the $\sum_{y'=1}^{2}$. To use our α and β parameters, which we defined as log probabilities, we will add in an awkward exp log in (13). This means we have an exp() function inside this equation, that we cannot get rid of. If V is large, the term $\sum_{j=1}^{V} X_{i,j}\beta_{j,y'}$ may be a large negative number, and exponentiating it may result in underflow. You will solve this problem in the `stable_log_sum` function you write in `src/utils.py`.

A second crucial difference is that our y_i variables are latent, and so we will estimate them using our $p(y_i \mid X_i, \alpha, \beta)$. You can connect this back to (5) by imagining that when we are given a fully-supervised dataset, that $p(y_i = y' \mid X_i, \alpha, \beta)$ is either 0 or 1 depending on whether y_i is in fact y' or not. When our y_i labels are latent, we need to predict a distribution over these y' labels. The challenge this introduces is that our likelihood now has three terms in it that are 'trapped' together inside the exp function: two depend on α, and two depend on β. This means we cannot compute derivatives directly, and will need to turn to the EM algorithm to optimize α and β.

Remember from class that the general form of the EM algorithm involves alternating between two steps: first we will find the *expectation* of the unobserved values given our current model parameters, then we will pretend those are the true values and use them to find (*maximize* for) better model parameters. We saw Gaussian Mixture Models (GMMs) as an example of the EM algorithm, where the first ("E") step involved assigning each data point to a cluster based on the current location of the clusters, and the second ("M") step involved updating the parameters of each cluster based on the points assigned to it.

For fully-unsupervised Naive Bayes, we will take a similar approach. We start by initializing our model parameters somehow, and then in the E-step we use those parameters to predict $p(y_i \mid X_i, \alpha, \beta)$ for every example i. These predicted probabilities are our latent variables. In the M-step, we treat those predicted probabilities as fixed, and use them to update our α and β values. We keep iterating between E and M steps until the algorithm converges. The algorithm is described in Figure 1.

5 Semi-supervised Naive Bayes

In your homework, you will consider a model that lies between the fully-supervised and fully-unsupervised settings. This means that we will have *some* examples i for which we see y_i, but others for which we do not. The approach to such data is simply a modified version of the EM algorithm we just introduced. If we have a label y_i, use it; if not, predict it using $p(y_i = y' \mid X_i, \alpha^t, \beta^t)$. Note that it is crucially important not to overwrite the true y_i with our E-step predictions. At each time t, $p(y_i = y' \mid X_i, \alpha^t, \beta^t)$ will *either* be a prediction based on our current α^t and β^t, or it will be the true y_i labels that were given to us at the beginning. The semi-supervised algorithm is defined in Figure 2; you should use this to guide your implementation of `fit()` in `src/naive_bayes_em.py`.

The likelihood for semi-supervised Naive Bayes is identical to 14, except that we have a slightly different definition for $p(y_i \mid X_i, \alpha, \beta)$. If we have a label y_i, we will use, saying $p(y_i = y' \mid X_i, \alpha, \beta)$ is 1 if y_i is in fact equal to y', or else 0. If we do not have a label y_i, then we will infer a distribution over possible values y' and treat it as a latent variable.

One final note: While this document does not mention smoothing, it is an important piece of the coding implementation. It is omitted here only to make the probabilistic calculations clearer.

Inputs:

- A dataset $\{X_i\}_{i=1}^N$, where X has shape $[N, V]$.
- A value K of the number of labels to consider; in this assignment, $K = 2$.
- A maximum number of iterations T.

Initialization:

- Initialize $\alpha^0 = \log p(y_i = y') = -\log K$ for all i and y'.
- Initialize $\beta^0 = \log p(w_j \mid y_i = y') = -\log V$ for all i, j, and y'.

Algorithm:
For $t = 1 \ldots T$ or until convergence,

E-step: For all i and y', compute:

$$p(y_i = y' \mid X_i, \alpha^t, \beta^t) = \frac{p(X_i \mid y_i, \beta^t) p(y_i \mid \alpha^t)}{\sum_{y'=1}^{K} p(X_i \mid y', \beta^t) p(y' \mid \alpha^t)} \tag{15}$$

$$\tag{16}$$

M-step: For all y' and j, compute:

$$\alpha_{y'}^{t+1} = \log \left(\frac{1}{N} \sum_{i=1}^{N} p(y_i = y' \mid X_i, \alpha^t, \beta^t) \right) \tag{17}$$

$$\beta_{j,y'}^{t+1} = \log \left(\frac{\sum_{i=1}^{N} X_{i,j}\, p(y_i = y' \mid X_i, \alpha^t, \beta^t)}{\sum_{i=1}^{N} \sum_{j=1}^{V} X_{i,j}\, p(y_i = y' \mid X_i, \alpha^t, \beta^t)} \right) \tag{18}$$

Figure 1: The EM algorithm for fully-unsupervised Naive Bayes. Note that (15) is essentially identical to (6); it will be easier using your α and β by using (9). Note the close similarity between (17) and (10) and between (18) and (11).

Inputs:

- A dataset $\{X_i, y_i\}_{i=1}^N$, where X has shape $[N, V]$ and y has shape $[N, 1]$; each y_i is either a label from the set$\{0, \ldots, K-1\}$, or can take a special value of "?" if the ith example has no label. In your homework, "?" will be represented as `np.nan`.
- A maximum number of iterations T.

Initialization:

- Initialize $\alpha^0 = \log p(y_i = y') = -\log K$ for all i and y'.
- Initialize $\beta^0 = \log p(w_j \mid y_i = y') = -\log V$ for all i, j, and y'.

Algorithm:
For $t = 1 \ldots T$ or until convergence,

E-step: For all i such that $y_i = ?$, for all y', compute:

$$p(y_i = y' \mid X_i, \alpha^t, \beta^t) = \frac{p(X_i \mid y_i, \beta^t) p(y_i \mid \alpha^t)}{\sum_{y'=1}^K p(X_i \mid y', \beta^t) p(y' \mid \alpha^t)} \tag{19}$$

Then, for all i such that $y_i \neq ?$, define

$$p(y_i = y' \mid X_i, \alpha^t, \beta^t) = \begin{cases} 1 & y' = y_i \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

M-step: For all y' and j, compute:

$$\alpha_{y'}^{t+1} = \log\left(\frac{1}{N}\sum_{i=1}^N p(y_i = y' \mid X_i, \alpha^t, \beta^t)\right) \tag{21}$$

$$\beta_{j,y'}^{t+1} = \log\left(\frac{\sum_{i=1}^N X_{i,j}\, p(y_i = y' \mid X_i, \alpha^t, \beta^t)}{\sum_{i=1}^N \sum_{j=1}^V X_{i,j}\, p(y_i = y' \mid X_i, \alpha^t, \beta^t)}\right) \tag{22}$$

Figure 2: The EM algorithm for semi-supervised Naive Bayes. Note that (17) is identical to (21) and (18) is identical to (22), except that $p(y_i \mid X_i, \alpha^t, \beta^t)$ is defined differently in the E-step.

A Supervised update derivation

You should not need to use this section to implement your homework. However, if you are interested in where the fully-supervised update rule comes from, you may find this section helpful. If you are interested in the theoretical motivation behind the EM algorithm, refer to the lecture slides or section 6 ("The EM algorithm in General Form") from the writeup that inspired this document, but note that it uses slightly different notation. To cover the math more generally below, we'll refer to the number of labels as K, even though the coding assumes $K = 2$.

To get started with deriving our supervised update rules, remember that $p(y_i = y) = \alpha_y$ and $p(w_j \mid y_i = y) = \beta_{j,y}$, and that our goal in (5) was to find $\{\alpha, \beta\}$ that maximize:

$$\mathcal{L}(X, y, \alpha, \beta) = \sum_{i=1}^{N} \log p(y_i \mid \alpha) + \sum_{i=1}^{N} \sum_{j=1}^{V} \log p(X_{i,j} \mid y_i, \beta) \tag{23}$$

Note that

$$\begin{aligned} \log p(y_i \mid \alpha) &= \log \left(\prod_{y'=1}^{K} \alpha_{y'}^{\mathbb{1}(y_i = y')} \right) \\ &= \mathbb{1}(y_i = y) \log \alpha_y \end{aligned} \tag{24}$$

and

$$\begin{aligned} \log p(X_{i,j} \mid y_i, \beta) &= X_{i,j} \log \prod_{y'=1}^{K} (\beta_{j,y'})^{\mathbb{1}(y_i = y')} \\ &= X_{i,j} \sum_{y'=1}^{K} \mathbb{1}(y_i = y') \log \beta_{j,y'} \\ &= X_{i,j} \mathbb{1}(y_i = y) \log \beta_{j,y} \end{aligned} \tag{25}$$

Note that for $p(X \mid y)$ and $p(y)$ to be valid probability distributions, our optimization of $\mathcal{L}$ must be subject to the equality constraints that $\sum_y \alpha_y = 1$ and, for all labels y, $\sum_{j=1}^{V} \beta_{j,y} = 1$. We can encode these constraints as Lagrange multipliers by optimizing:

$$\begin{aligned}
&\mathcal{L}'(X, y, \alpha, \beta) \\
&\quad = \mathcal{L}(X, y, \alpha, \beta) - \lambda_\alpha\big(-1 + \sum_y \alpha_y\big) - \sum_y \lambda_{\beta,y}\big(-1 + \sum_{j=1}^{V} \beta_{j,y}\big) \\
&\quad = \sum_{i=1}^{N} \log p(y_i \mid \alpha) - \lambda_\alpha\big(-1 + \sum_y \alpha_y\big) \\
&\qquad + \sum_{i=1}^{N}\sum_{j=1}^{V} \log p(X_{i,j} \mid y_i, \beta) - \sum_y \lambda_{\beta,y}\big(-1 + \sum_{j=1}^{V} \beta_{j,y}\big) \\
&\quad = \sum_{i=1}^{N} \mathbb{1}(y_i = y) \log \alpha_y - \lambda_\alpha\big(-1 + \sum_y \alpha_y\big) \qquad (26) \\
&\qquad + \sum_{i=1}^{N}\sum_{j=1}^{V} X_{i,j}\mathbb{1}(y_i = y) \log \beta_{j,y} - \sum_y \lambda_{\beta,y}\big(-1 + \sum_{j=1}^{V} \beta_{j,y}\big) \qquad (27)
\end{aligned}$$

Noting that (26) only depends on α and λ_α and not β nor $\lambda_{\beta,y}$, we can compute the derivative with respect to α_y (one element of α) as:

$$\begin{aligned}
&\frac{\partial}{\partial \alpha_y}\mathcal{L}'(X, y, \alpha, \beta) \\
&\quad = \frac{\partial}{\partial \alpha_y}\left[\left(\sum_{i=1}^{N} \mathbb{1}(y_i = y) \log \alpha_y\right) - \lambda_\alpha\left(-1 + \sum_{y'=1}^{K} \alpha_{y'}\right)\right] \\
&\quad = \frac{1}{\alpha_y}\left(\sum_{i=1}^{N} \mathbb{1}(y_i = y)\right) - \lambda_\alpha
\end{aligned}$$

Therefore, setting the derivative to 0 and solving, we get:

$$\begin{aligned}
&\frac{\partial}{\partial \alpha_y}\mathcal{L}'(X, y, \alpha, \beta) = 0 \\
&\qquad \Rightarrow \alpha_y = \frac{1}{\lambda_\alpha}\sum_{i=1}^{N} \mathbb{1}(y_i = y) \qquad (28)
\end{aligned}$$

Similarly, noting that (27) only depends on β and $\lambda_{\beta,y}$, we can compute the derivative with respect to $\beta_{j,y}$ as:

$$\begin{aligned}
&\frac{\partial}{\partial \beta_{j,y}}\mathcal{L}'(X, y, \alpha, \beta) \\
&\quad = \frac{\partial}{\partial \beta_{j,y}}\left[\left(\sum_{i=1}^{N}\sum_{j=1}^{V} X_{i,j}\mathbb{1}(y_i = y) \log \beta_{j,y}\right) - \sum_{y'=1}^{K} \lambda_{\beta,y'}\left(-1 + \sum_{j=1}^{V} \beta_{j,y'}\right)\right] \\
&\quad = \frac{1}{\beta_{j,y}}\left(\sum_{i=1}^{N} X_{i,j}\mathbb{1}(y_i = y)\right) - \lambda_{\beta,y} \qquad (29)
\end{aligned}$$

Therefore, setting the derivative to 0 and solving, we get:

$$\begin{aligned}\frac{\partial}{\partial \beta_{j,y}} \mathcal{L}'(X, y, \alpha, \beta) = 0 & \\ \Rightarrow \beta_{j,y} = \frac{1}{\lambda_{\beta,y}} \sum_{i=1}^{N} X_{i,j} \mathbb{1}(y_i = y) & \qquad (30)\end{aligned}$$

To find a solution for the Lagrangian, we must also find the derivative with respect to the λ_α parameter:

$$\begin{aligned}&\frac{\partial}{\partial \lambda_\alpha} \mathcal{L}'(X, y, \alpha, \beta) \\ &\quad = \frac{\partial}{\partial \lambda_\alpha} \left[\left(\sum_{i=1}^{N} \mathbb{1}(y_i = y) \log \alpha_y \right) - \lambda_\alpha \left(-1 + \sum_{y'=1}^{K} \alpha_{y'} \right) \right] \\ &\quad = 1 - \sum_{y'=1}^{K} \alpha_{y'} \qquad (31)\end{aligned}$$

Setting to zero implies the probability distribution constraint:

$$\begin{aligned}\frac{\partial}{\partial \lambda_\alpha} \mathcal{L}'(X, y, \alpha, \beta) = 0 & \\ \Rightarrow \sum_{y'=1}^{K} \alpha_{y'} = 1 & \qquad (32)\end{aligned}$$

Repeating this for the $\lambda_{\beta,y}$ parameters, we get:

$$\begin{aligned}&\frac{\partial}{\partial \lambda_{\beta,y}} \mathcal{L}'(X, y, \alpha, \beta) \\ &\quad = \frac{\partial}{\partial \lambda_{\beta,y}} \left[\left(\sum_{i=1}^{N} \sum_{j=1}^{V} X_{i,j} \mathbb{1}(y_i = y) \log \beta_{j,y} \right) - \sum_{y'=1}^{K} \lambda_{\beta,y'} \left(-1 + \sum_{j=1}^{V} \beta_{j,y'} \right) \right] \\ &\quad = \frac{\partial}{\partial \lambda_{\beta,y}} - \sum_{y'=1}^{K} \lambda_{\beta,y'} \left(-1 + \sum_{j=1}^{V} \beta_{j,y'} \right) \\ &\quad = 1 - \sum_{j=1}^{V} \beta_{j,y} \qquad (33)\end{aligned}$$

Setting the derivative to zero implies the desired constraint:

$$\begin{aligned}\frac{\partial}{\partial \lambda_{\beta,y}} \mathcal{L}'(X, y, \alpha, \beta) = 0 & \\ \Rightarrow \sum_{j=1}^{V} \beta_{j,y} = 1 & \qquad (34)\end{aligned}$$

To solve for λ_α, note that combining (28) and (32) gives us:

$$\begin{aligned}
1 &= \sum_{y'=1}^{K} \alpha_{y'} \\
&= \sum_{y'=1}^{K} \frac{1}{\lambda_\alpha} \sum_{i=1}^{N} \mathbb{1}(y_i = y') \\
&= \frac{1}{\lambda_\alpha} \sum_{i=1}^{N} \sum_{y'=1}^{K} \mathbb{1}(y_i = y') \\
&= \frac{1}{\lambda_\alpha} \sum_{i=1}^{N} 1 \\
&= \frac{N}{\lambda_\alpha}
\end{aligned}$$

And therefore plugging $\lambda_\alpha = N$ into (28) gives us the update we saw in (10).

To solve for $\lambda_{\beta,y}$, take a similar approach using (30) and (34):

$$\begin{aligned}
1 &= \sum_{j=1}^{V} \beta_{j,y} \\
&= \sum_{j=1}^{V} \frac{1}{\lambda_{\beta,y}} \sum_{i=1}^{N} X_{i,j} \mathbb{1}(y_i = y) \\
&= \frac{1}{\lambda_{\beta,y}} \sum_{i=1}^{N} \sum_{j=1}^{V} X_{i,j} \mathbb{1}(y_i = y)
\end{aligned}$$

So in plain language, $\lambda_{\beta,y} = \sum_{i=1}^{N} \sum_{j=1}^{V} X_{i,j} \mathbb{1}(y_i = y)$ is the total number of words among documents with the label $y_i = y$. Plugging this definition of $\lambda_{\beta,y}$ into (30) gives us the update rule we saw in (11).