CS-TR-4956
UMIACS-TR-2010-04
LAMP-TR-153

June 2010

# GIBBS SAMPLING FOR THE UNINITIATED

Philip Resnik

Department of Linguistics
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
*resnik AT umd.edu*

Eric Hardisty

Department of Computer Science
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
*hardisty AT cs.umd.edu*

**Abstract**

This document is intended for computer scientists who would like to try out a Markov Chain Monte Carlo (MCMC) technique, particularly in order to do inference with Bayesian models on problems related to text processing. We try to keep theory to the absolute minimum needed, though we work through the details much more explicitly than you usually see even in "introductory" explanations. That means we've attempted to be ridiculously explicit in our exposition and notation.

After providing the reasons and reasoning behind Gibbs sampling (and at least nodding our heads in the direction of theory), we work through an example application in detail—the derivation of a Gibbs sampler for a Naïve Bayes model. Along with the example, we discuss some practical implementation issues, including the integrating out of continuous parameters when possible. We conclude with some pointers to literature that we've found to be somewhat more friendly to uninitiated readers.

Note: as of June 3, 2010 we have corrected some small errors in the original April 2010 report.

**Keywords: Gibbs sampling, Markov Chain Monte Carlo, naïve Bayes, Bayesian inference, tutorial**

# Gibbs Sampling for the Uninitiated

Philip Resnik
Department of Linguistics and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742 USA
resnik AT umd.edu

Eric Hardisty
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742 USA
hardisty AT cs.umd.edu

**Abstract**

This document is intended for computer scientists who would like to try out a Markov Chain Monte Carlo (MCMC) technique, particularly in order to do inference with Bayesian models on problems related to text processing. We try to keep theory to the absolute minimum needed, though we work through the details much more explicitly than you usually see even in "introductory" explanations. That means we've attempted to be ridiculously explicit in our exposition and notation.

After providing the reasons and reasoning behind Gibbs sampling (and at least nodding our heads in the direction of theory), we work through an example application in detail—the derivation of a Gibbs sampler for a Naïve Bayes model. Along with the example, we discuss some practical implementation issues, including the integrating out of continuous parameters when possible. We conclude with some pointers to literature that we've found to be somewhat more friendly to uninitiated readers.

## 1 Introduction

Markov Chain Monte Carlo (MCMC) techniques like Gibbs sampling provide a principled way to approximate the value of an integral.

### 1.1 Why integrals?

Ok, stop right there. Many computer scientists, including a lot of us who focus in natural language processing, don't spend a lot of time with integrals. We spend most of our time and energy in a world of discrete events. (The word *bank* can mean (1) a financial institution, (2) the side of a river, or (3) tilting an airplane. Which meaning was intended, based on the words that appear nearby?) Take a look at Manning and Schuetze [14], and you'll see that the probabilistic models we use tend to involve sums, not integrals (the Baum-Welch algorithm for HMMs, for example). So we have to start by asking: why and when do we care about integrals?

One good answer has to do with probability estimation.[1] Numerous computational methods involve estimating the probabilities of alternative discrete choices, often in order to pick the single most probable choice. As one example, the language model in an automatic speech recognition (ASR) system estimates the probability of the next word given the previous context. As another example, many spam blockers use features of the e-mail message (like the word *Viagra*, or the phrase *send this message to all your friends*) to predict the probability that the message is spam.

Sometimes we estimate probabilities by using *maximimum likelihood estimation* (MLE). To use a standard example, if we are told a coin may be unfair, and we flip it 10 times and see HHHHTTTTTT (H=heads, T=tails), it's conventional to estimate the probability of heads for the next flip as 0.4. In practical terms, MLE amounts to counting and then normalizing so that the probabilities sum to 1.

[1] This subsection is built around the very nice explication of Bayesian probability estimation by Heinrich [7].

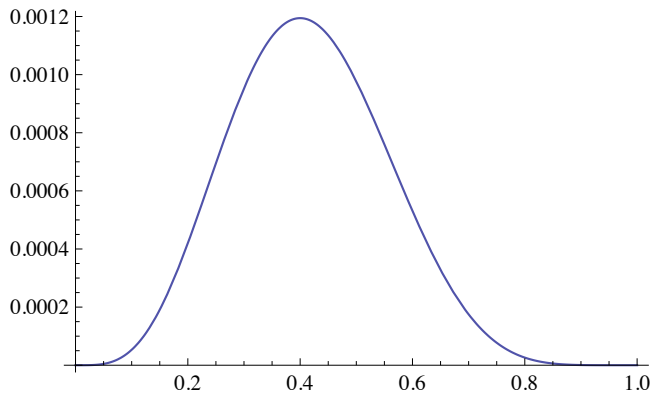


Figure 1: Probability of generating the coin-flip sequence HHHHTTTTTT, using different values for $P(\text{heads})$ on the $x$-axis. The value that maximizes the probability of the observed sequence, 0.4, is the maximum likelihood estimate (MLE).

$$\frac{\text{count(H)}}{\text{count(H)} + \text{count(T)}} = \frac{4}{10} = 0.4 \tag{1}$$

Formally, MLE produces the choice *most likely to have generated the observed data.*

In this case, the most natural model $\mu$ has just a single parameter, $\pi$, namely the probability of heads (see Figure 1).[2] Letting $\mathcal{X} = \text{HHHHTTTTTT}$ represent the observed data, and $y$ the outcome of the next coin flip, we estimate

$$\begin{aligned} \tilde{\pi}_{MLE} &= \operatorname*{argmax}_{\pi} P(\mathcal{X}|\pi) && (2) \\ P(y|\mathcal{X}) &\approx P(y|\tilde{\pi}_{MLE}) && (3) \end{aligned}$$

On the other hand, sometimes we estimate probabilities using *maximum a posteriori* (MAP) estimation. A MAP estimate is the choice that is *most likely given the observed data.* In this case,

$$\begin{aligned} \tilde{\pi}_{MAP} &= \operatorname*{argmax}_{\pi} P(\pi|\mathcal{X}) \\ &= \operatorname*{argmax}_{\pi} \frac{P(\mathcal{X}|\pi)P(\pi)}{P(\mathcal{X})} \\ &= \operatorname*{argmax}_{\pi} P(\mathcal{X}|\pi)P(\pi) && (4) \\ P(y|\mathcal{X}) &\approx P(y|\tilde{\pi}_{MAP}) && (5) \end{aligned}$$

In contrast to MLE, MAP estimation applies Bayes's Rule, so that our estimate (4) can take into account *prior knowledge* about what we expect $\pi$ to be in the form of a prior probability distribution $P(\pi)$.[3] So,

[2]Specifically, $\mu$ models each choice as a Bernoulli trial, and the probability of generating exactly this heads-tails sequence for a given $\pi$ is $\pi^4(1-\pi)^6$. If you type `Plot[p^4(1-p)^6,{p,0,1}]` into Wolfram Alpha, you get Figure 1, and you can immediately see that the curve tops out, i.e. the probability of generating the sequence is highest, exactly when $p = 0.4$. Confirm this by entering `derivative of p^4(1-p)^6` and you'll get $\frac{2}{5} = 0.4$ as the maximum. Thanks to Kevin Knight for pointing out how easy all this is using Wolfram Alpha. Also see discussion in Heinrich [7], Section 2.1.

[3]We got to (4) from the desired posterior probability by applying Bayes's Rule and then ignoring the denominator since the argmax doesn't depend on it.

for example, we might believe that the coin flipper is a scrupulously honest person, and choose a prior distribution that is biased in favor of $\pi = 0.5$. The more heavily biased that prior distribution is, the more evidence it will take to shake our pre-existing belief that the coin is fair.[4]

Now, MLE and MAP estimates are both giving us the *best* estimate, according to their respective definitions of "best." But notice that using a single estimate — whether it's $\tilde{\pi}_{MLE}$ or $\tilde{\pi}_{MAP}$ — throws away information. In principle, $\pi$ could have *any* value between 0 and 1; might we not get better estimates if we took the whole distribution $P(\pi|\mathcal{X})$ into account, rather than just a single estimated value for $\pi$? If we do that, we're making use of all the information about $\pi$ that we can wring from the observed data, $\mathcal{X}$.

The way to take advantage of all that information is to calculate an *expected value* rather than an estimate using the single best guess for $\pi$. Recall that the expected value of a function $f(z)$, when $z$ is a discrete variable, is

$$E[f(z)] = \sum_{z \in \mathcal{Z}} f(z)p(z). \tag{6}$$

Here $\mathcal{Z}$ is the set of discrete values $z$ can take, and $p(z)$ is the probability distribution over possible values for $z$. If $z$ is a continuous variable, the expected value is an integral rather than a sum:

$$E[f(z)] = \int f(z)p(z)\,\mathrm{d}z. \tag{7}$$

For our example, $z = \pi$, the function $f$ we're interested in is $f(z) = P(y|\pi)$, and the distribution over which we're taking the expectation is $P(\pi|\mathcal{X})$, i.e. the whole distribution over possible values of $\pi$ given that we've observed $\mathcal{X}$. That gives us the following expected value for the posterior probability of $y$ given $\mathcal{X}$:

$$P(y|\mathcal{X}) = \int P(y|\pi)P(\pi|\mathcal{X})\,\mathrm{d}\pi \tag{8}$$

where Bayes's Rule defines

$$P(\pi|\mathcal{X}) = \frac{P(\mathcal{X}|\pi)P(\pi)}{P(\mathcal{X})} = \frac{P(\mathcal{X}|\pi)P(\pi)}{\int_\pi P(\mathcal{X}|\pi)P(\pi)\,\mathrm{d}\pi}. \tag{9}$$

Notice that, unlike (3) and (5), Equation (8) defines the posterior using a true equality, not an approximation. It takes fully into account our prior beliefs about what the value of $\pi$ will be, along with the interaction of those prior beliefs with observed evidence $\mathcal{X}$.

Equations (8) and (9) provide one compelling answer to the question we started with. Why should even discrete-minded computer scientists care about integrals? Because even when the probability space is discrete, we often care about good estimates of posterior probabilities. Computing integrals can help us improve the parameter estimates in our models.[5]

[4] See http://www.math.uah.edu/STAT/objects/experiments/BetaCoinExperiment.xhtml for a nice applet that lets you explore this idea. If you set $a = b = 10$, you get a prior strongly biased toward 0.5, and it's hard to move the posterior too far from that value even if you generate observed heads with probability $p = 0.8$. If you set $a = b = 2$, there's still a bias toward 0.5 but it's much easier to move the posterior off that value. As a second pointer, see some nice, self-contained slides at http://www.cs.cmu.edu/∼lewicki/cp-s08/Bayesian-inference.pdf.

[5] Chris Dyer (personal communication) points out you don't have to be doing Bayesian estimation to care about expected values. For example, better ways to compute expected values can be useful in the E step of expectation-maximization algorithms, which give you maximum likelihood estimates for models with latent variables. He also points out that for many models, Bayesian parameter estimation can be a whole lot easier to implement than EM. The widely used GIZA++ implementation of IBM Model 3 (a probabilistic model used in statistical machine translation [12]) contains 2186 lines of code; Chris implemented a Gibbs sampler for Model 3 in 67 lines. On a related note, Kevin Knight's excellent "Bayesian Inference with Tears: A tutorial workbook for natural language researchers" [9] was written with goals very similar to our own, but from an almost completely complementary angle: he emphasizes conceptual connections to EM algorithms and focuses on the kinds of structured problems you tend to see in natural language processing.

### 1.2 Why sampling?

The trouble with integrals, of course, is that they can be very difficult to calculate. The methods we learned in calculus class are fine for classroom exercises, but often cannot be applied to interesting problems in the real world. Indeed, analytical solutions to (8) and the denominator of (9) might be impossible to obtain, so we might not be able to determine the exact form of $P(\pi|\mathcal{X})$. Gibbs sampling allows us to sample from a distribution that asymptotically follows $P(\pi|\mathcal{X})$ without having to explicitly calculate the integrals.

#### 1.2.1 Monte Carlo: a circle, a square, and a bag of rice

Gibbs Sampling is an instance of a Markov Chain Monte Carlo technique. Let's start with the "Monte Carlo" part. You can think of Monte Carlo methods as algorithms that help you obtain a desired value by performing simulations involving probabilistic choices. As a simple example, here's a cute, low-tech Monte Carlo technique for estimating the value of $\pi$ (the ratio of a circle's circumference to its diameter).[6]

Draw a perfect square on the ground. Inscribe a circle in it — i.e. the circle and the square are centered in exactly the same place, and the circle's diameter has length identical to the side of the square. Now take a bag of rice, and scatter the grains uniformly at random inside the square. Finally, count the total number of grains of rice inside the circle (call that C), and inside the square (call that S).

You scattered rice at random. Assuming you managed to do this pretty uniformly, the ratio between the circle's grains and the square's grains (which include the circle's) should approximate the ratio between the area of the circle and the area of the square, so

$$\frac{C}{S} \approx \frac{\pi(\frac{d}{2})^2}{d^2}. \tag{10}$$

Solving for $\pi$, we get $\pi \approx \frac{4C}{S}$.

You may not have realized it, but we just solved a problem by approximating the values of integrals. The true area of the circle, $\pi(\frac{d}{2})^2$, is the result of summing up an infinite number of infinitessimally small points; similarly for the the true area $d^2$ of the square. The more grains of rice we use, the better our approximation will be.

#### 1.2.2 Markov Chains: walking the right walk

In the circle-and-square example, we saw the value of sampling involving a uniform distribution, since the grains of rice were distributed uniformly within the square. Returning to the problem of computing expected values, recall that we're interested in $E_{p(x)}[f(x)]$ (equation 7), where we'll assume that the distribution $p(x)$ is *not* uniform and, in fact, not easy to work with analytically.

Figure 2 provides an example $f(z)$ and $p(z)$ for illustration. Conceptually, the integral in equation (7) sums up $f(z)p(z)$ over infinitely many values of $z$. But rather than touching each point in the sum exactly once, here's another way to think about it: if you sample $N$ points $z^{(0)}, z^{(1)}, z^{(2)}, \ldots, z^{(N)}$ at random from the probability density $p(z)$, then

$$E_{p(z)}[f(z)] = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} f(z^{(t)}). \tag{11}$$

That looks a lot like a kind of averaged value for $f$, which makes a lot of sense since in the discrete case (equation 6) the expected value is nothing but a weighted average, where the weight for each value of $z$ is its probability.

Notice, though, that the value in the sum is just $f(z^{(t)})$, not $f(z^{(t)})p(z^{(t)})$ as in the integral in equation (7). Where did the $p(z)$ part go? Intuitively, if we're sampling according to $p(z)$, and count$(z)$ is the number of

[6] We're elaborating on the introductory example at `http://en.wikipedia.org/wiki/Monte_Carlo_method`.

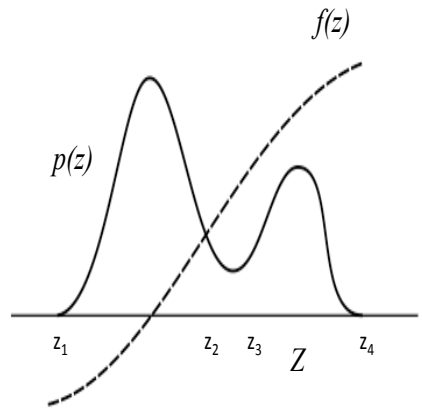


Figure 2: Example of computing expectation $E_{p(z)}[f(z)]$. (This figure was adapted from page 7 of the handouts for Chris Bishop's presentation "NATO ASI: Learning Theory and Practice", Leuven, July 2002, http://research.microsoft.com/en-us/um/people/cmbishop/downloads/bishop-nato-2.pdf )

times we observe $z$ in the sample, then $\frac{1}{N}\text{count}(z)$ approaches $p(z)$ as $N \to \infty$. So the $p(z)$ is implicit in the way the samples are drawn.[7]

Looking at equation (11), it's clear that we can get an *approximate* value by sampling only a finite number of times, $T$:

$$E_{p(z)}[f(z)] \approx \frac{1}{T}\sum_{t=1}^{T} f(z^{(t)}). \tag{12}$$

Progress! Now we have a way to approximate the integral. The remaining problem is this: how do we sample $z^{(0)}, z^{(1)}, z^{(2)}, \ldots, z^{(T)}$ according to $p(z)$?

There are a whole variety of ways to go about this; e.g., see Bishop [2] Chapter 11 for discussion of rejection sampling, adaptive rejection sampling, adaptive rejection Metropolis sampling, importance sampling, sampling-importance-sampling,... For our purposes, though, the key idea is to think of $z$'s as points in a state space, and find ways to "walk around" the space — going from $z^{(0)}$ to $z^{(1)}$ to $z^{(2)}$ and so forth — so that the likelihood of visiting any point $z$ is proportional to $p(z)$. Figure 2 illustrates the reasoning visually: walking around values for $Z$, we want to spend our time adding values $f(z)$ to the sum when $p(z)$ is large, e.g. devoting more attention to the space between $z_1$ and $z_2$, as opposed to spending time in less probable portions of the space like the part between $z_2$ and $z_3$.

A walk of this kind can be characterized abstractly as follows:

```
z^(0) := a random initial point
for t = 1 to T do
    z^(t+1) := g(z^(t))
end for
```

[7] Okay — we're playing a little fast and loose here by talking about counts: with $Z$ continuous, we're not going to see two identical samples $z^{(i)} = z^{(j)}$, so it doesn't really make sense to talk about counting how many times a value was seen. But we did say "intuitively", right?

Here $g$ is a function that makes a probabilistic choice about what state to go to next according to an explicit or implicit transition probability $P_{\text{trans}}(z^{(t+1)}|z^{(0)}, z^{(1)}, \ldots, z^{(t)})$.[8]

The part about probabilistic choices makes this a Monte Carlo technique. What will make it a *Markov Chain* Monte Carlo technique is defining things so that the next state you visit, $z^{(t+1)}$, depends *only* on the current state $z^{(t)}$. That is,

$$P_{\text{trans}}(z^{(t+1)}|z^{(0)}, z^{(1)}, \ldots, z^{(t)}) = P_{\text{trans}}(z^{(t+1)}|z^{(t)}). \tag{13}$$

For you language folks, this is precisely the same idea as modeling word sequences using a bigram model, where here we have states $z$ instead of having words $w$.

We included the subscript *trans* in our notation, so that $P_{\text{trans}}$ can be read as "transition probability", in order to emphasize that these are state-to-state transition probabilities in a (first-order) Markov model. The heart of Markov Chain Monte Carlo methods is designing $g$ so that the probability of visiting a state $z$ will turn out to be $p(z)$, as desired. This can be accomplished by guaranteeing that the chain, as defined by the transition probabilities $P_{\text{trans}}$, meets certain conditions. Gibbs sampling is one algorithm that meets those conditions.[9]

### 1.2.3 The Gibbs sampling algorithm

Gibbs sampling is applicable in situations where $Z$ has at least two dimensions, i.e. each point $z$ is really $z = \langle z_1, \ldots, z_k \rangle$, with $k > 1$. The basic idea in Gibbs sampling is that, rather than probabilistically picking the next state all at once, you make a separate probabilistic choice for each of the $k$ dimensions, where each choice depends on the other $k-1$ dimensions.[10] That is, the probabilistic walk through the space proceeds as follows:

---

$z^{(0)} := \langle z_1^{(0)}, \ldots, z_k^{(0)} \rangle$
**for** $t = 1$ to $T$ **do**
  **for** $i = 1$ to $k$ **do**
    $z_i^{(t+1)} \sim P(Z_i | z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \ldots, z_k^{(t)})$
  **end for**
**end for**

---

Note that we can obtain the distribution we are sampling from by using the definition of conditional probability:

$$P(Z_i | z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \ldots, z_k^{(t)}) = \frac{P(z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_i^{(t)}, z_{i+1}^{(t)}, \ldots, z_k^{(t)})}{P(z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \ldots, z_k^{(t)})}. \tag{14}$$

Notice that the only difference between the numerator and the denominator is that the numerator is the full joint probability, including $z_i^{(t)}$, whereas $z_i^{(t)}$ is missing in the denominator. This will be important later.

One full execution of the inner loop and you've just computed your new point $z^{(t+1)} = g(z^{(t)}) = \langle z_1^{(t+1)}, \ldots, z_k^{(t+1)} \rangle$.

You can think of each dimension $z_i$ as corresponding to a parameter or variable in your model. Using equation (14), we sample the new value for each variable according to its distribution based on the values of all the *other* variables. During this process, new values for the variables are used *as soon as you obtain them*. For the case of three variables:

- The new value of $z_1$ is sampled conditioning on the old values of $z_2$ and $z_3$.

[8] We're deliberately staying at a high level here. In the bigger picture, $g$ might consider and reject one or more states before finally deciding to accept one and return it as the value for $z^{(t+1)}$. See discussions of the Metropolis-Hastings algorithm, e.g. Bishop [2], Section 11.2.

[9] We told you we were going to keep theory to a minimum, didn't we?

[10] There are, of course, variations on this basic scheme. For example, "blocked sampling" groups the variables into $b < k$ blocks and the variables in each block are sampled together based on the other $b-1$ blocks.

- The new value of $z_2$ is sampled conditioning on the *new* value of $z_1$ and the *old* value of $z_3$.
- The new value of $z_3$ is sampled conditioning on the *new* values of $z_1$ and $z_2$.

### 1.3 The remainder of this document

So, there you have it. Gibbs sampling makes it possible to obtain samples from probability distributions without having to explicitly calculate the values for their marginalizing integrals, e.g. computing expected values, by defining a conceptually straightforward approximation. This approximation is based on the idea of a probabilistic walk through a state space whose dimensions correspond to the variables or parameters in your model.

Trouble is, from what we can tell, most descriptions of Gibbs sampling pretty much stop there (if they've even gone into that much detail). To someone relatively new to this territory, though, that's not nearly far enough to figure out how to *do* Gibbs sampling. How exactly do you implement the "sampling from the following distribution" part at the heart of the algorithm (equation 14) for your particular model? How do you deal with continuous parameters in your model? How do you actually *generate* the expected values you're ultimately interested in (e.g. equation 8), as opposed to just doing the probabilistic walk for $T$ iterations?

Just as the first part of this document aimed at explaining *why*, the remainder aims to explain *how*. In Section 2, we take a very simple probabilistic model — Naïve Bayes — and describe in considerable (painful?) detail how to construct a Gibbs sampler for it. This includes two crucial things, namely how to employ conjugate priors and how to actually sample from conditional distributions per equation (14).

In Section 3 we discuss how to actually obtain values from a Gibbs sampler, as opposed to merely watching it walk around the state space. (Which might be entertaining, but wasn't really the point.) Our discussion includes convergence and burn-in, auto-correlation and lag, and other practical issues.

In Section 4 we conclude with pointers to other things you might find it useful to read, as well as an invitation to tell us how we could make this document more accurate or more useful.

## 2 Deriving a Gibbs Sampler for a Naïve Bayes Model

In this section we consider Naive Bayes models.[11] Let's assume that items of interest are documents, that the features under consideration are the words in the document, and that the document-level class variable we're trying to predict is a sentiment label whose value is either 0 or 1. For ease of reference, we present our notation in Figure 3. Figure 4 describes the model as a "plate diagram", to which we will refer when describing the model.

### 2.1 Modeling How Documents are Generated

We represent each document as a bag of words. Given an unlabeled document $\mathbf{W}_j$, our goal is to pick the best label $\mathrm{L}_j = 0$ or $\mathrm{L}_j = 1$. Sometimes we will refer to 0 and 1 as *classes* instead of labels. In further discussion it will be convenient for us to refer to the sets of documents sharing the same label, so for notational convenience we define the sets $\mathbb{C}_0 = \{\mathbf{W}_j | \mathrm{L}_j = 0\}$ and $\mathbb{C}_1 = \{\mathbf{W}_j | \mathrm{L}_j = 1\}$. The usual treatment of these models is to equate "best" with "most probable", and therefore our goal is to choose the label $\mathrm{L}_j$ for $\mathbf{W}_j$ that maximizes $P(\mathrm{L}_j|\mathbf{W}_j)$. Applying Bayes's Rule,

$$\begin{aligned}
\mathrm{L}_j = \operatorname*{argmax}_L P(L|\mathbf{W}_j) &= \operatorname*{argmax}_L \frac{P(\mathbf{W}_j|L)P(L)}{P(\mathbf{W}_j)} \\
&= \operatorname*{argmax}_L P(\mathbf{W}_j|L)P(L),
\end{aligned}$$

where the denominator $P(\mathbf{W}_j)$ is omitted because it does not depend on $L$.

This application of Bayes's rule (the "Bayes" part of "Naïve Bayes") allows us to think of the model in terms of a "generative story" that accounts for how documents are created. According to that story, we

[11] We assume the reader is familiar with Naive Bayes. For a refresher, see [14].

| | |
|---|---|
| $V$ | number of words in the vocabulary. |
| $N$ | number of documents in the corpus. |
| $\gamma_{\pi 1}$, $\gamma_{\pi 0}$ | hyperparameters of the Beta distribution. |
| $\boldsymbol{\gamma}_\theta$ | hyperparameter vector for the multinomial prior. |
| $\gamma_{\theta i}$ | pseudocount for word $i$. |
| $\mathbb{C}_x$ | set of documents labeled $x$. |
| $\mathbb{C}$ | the set of all documents. |
| $C_0$ $(C_1)$ | number of documents labeled 0 (1). |
| $\mathbf{W}_j$ | document $j$'s frequency distribution. |
| $\mathrm{W}_{ji}$ | frequency of word $i$ in document $j$. |
| $\mathbf{L}$ | vector of document labels. |
| $\mathrm{L}_j$ | label for document $j$. |
| $R_j$ | number of words in document $j$. |
| $\theta_i$ | probability of word $i$. |
| $\theta_{x,i}$ | probability of word $i$ from the distribution of class $x$. |
| $\mathcal{N}_{\mathbb{C}_x}(i)$ | number of times word $i$ occurs in the set of all documents labeled $x$. |
| $\mathbb{C}^{(-j)}$ | set of all documents *except* $\mathbf{W}_j$ |
| $\mathbf{L}^{(-j)}$ | vector of all document labels *except* $\mathrm{L}_j$ |
| $C_0^{(-j)}$ $(C_1^{(-j)})$ | number of documents labeled 0 (1) *except* for $\mathbf{W}_j$ |
| $\boldsymbol{\mu}$ | set of hyperparameters $\langle \gamma_{\pi 1}, \gamma_{\pi 0}, \boldsymbol{\gamma}_\theta \rangle$ |

Figure 3: Notation.

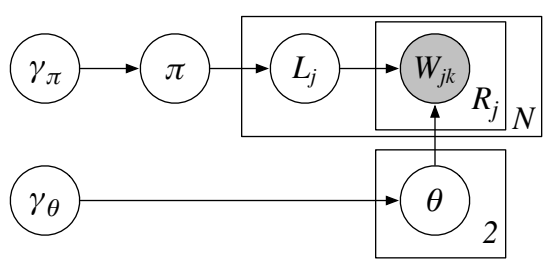


Figure 4: Naive Bayes plate diagram

first pick the class label of the document, $\mathrm{L}_j$; our model will assume that's done by flipping a coin whose probability of heads is some value $\pi = P(\mathrm{L}_j = 1)$. We can express this a little more formally as

$$\mathrm{L}_j \sim \mathrm{Bernoulli}(\pi).$$

Then, for every one of the $R_j$ word positions in the document, we pick a word $w_i$ independently by sampling randomly according to a probability distribution over words. Which probability distribution we use is based on the label $\mathrm{L}_j$ of the document, so we'll write them as $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$. Formally one would describe the creation of document $j$'s bag of words as

$$\mathbf{W}_j \sim \mathrm{Multinomial}(R_j, \boldsymbol{\theta}). \tag{15}$$

The assumption that the words are chosen independently is the reason we call the model "naïve".

Notice that logically speaking, it made sense to describe the model in terms of two separate probability distributions, $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$, each one being a simple unigram distribution. The notation in (15) doesn't explicitly show that what happens in generating $\mathbf{W}_j$ depends on whether $\mathrm{L}_j$ was 0 or 1. Unfortunately, that notational choice seems to be standard, even though it's less transparent.[12] We indicate the existence of two $\theta$s by including the 2 in the lower rectangle of Figure 4, but many plate diagrams in the literature would not. Another, perhaps clearer way to describe the process would be

$$\mathbf{W}_j \sim \mathrm{Multinomial}(R_j, \boldsymbol{\theta}_{\mathrm{L}_j}). \tag{16}$$

And that's it: our "generative story" for the creation of a whole set of labeled documents $\langle \mathbf{W}_n, \mathrm{L}_n \rangle$, according to the Naïve Bayes model, is that this simple document-level generative story gets repeated $N$ times, as indicated by the $N$ in the upper rectangle in Figure 4.

### 2.2 Priors

Well, ok, that's not completely it. Where did $\pi$ come from? Our generative story is going to assume that before this whole process began, we also picked $\pi$ randomly. Specifically we'll assume that $\pi$ is sampled from a Beta distribution with parameters $\gamma_{\pi 1}$ and $\gamma_{\pi 0}$. These are referred to as *hyperparameters* because they are parameters of a prior, which is itself used to pick parameters of the model. In Figure 4 we represent these two hyperparameters as a single two-dimensional vector $\boldsymbol{\gamma_\pi} = \langle \gamma_{\pi 1}, \gamma_{\pi 0} \rangle$. When $\gamma_{\pi 1} = \gamma_{\pi 0} = 1$, $Beta(\gamma_{\pi 1}, \gamma_{\pi 0})$ is just a uniform distribution, which means that any value for $\pi$ is equally likely. For this reason we call $Beta(1, 1)$ an "uninformed prior".

Similarly, where do $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ come from? Just as the Beta distribution can provide an uninformed prior for a distribution making a two-way choice, the Dirichlet distribution can provide an uninformed prior for $V$-way choices, where $V$ is the number of words in the vocabulary. Let $\boldsymbol{\gamma}_\theta$ be a $V$-dimensional vector where the value of every dimension equals 1. If $\boldsymbol{\theta}_0$ is sampled from $\mathrm{Dirichlet}(\boldsymbol{\gamma}_\theta)$, every probability distribution over words will be equally likely. Similarly, we'll assume $\boldsymbol{\theta}_1$ is sampled from $\mathrm{Dirichlet}(\boldsymbol{\gamma}_\theta)$.[13] Formally

$$\pi \sim \mathrm{Beta}(\boldsymbol{\gamma_\pi})$$

$$\boldsymbol{\theta} \sim \mathrm{Dirichlet}(\boldsymbol{\gamma}_\theta)$$

Choosing the Beta and Dirichlet distributions as priors for binomial and multinomial distributions, respectively, helps the math work out cleanly. We'll discuss this in more detail in Section 2.4.2.

[12] The actual basis for removing the subscripts on the parameters $\theta$ is that we assume the data from one class is independent of the parameter estimation of all the other classes, so essentially when we derive the probability expressions for one class the others look the same [19].

[13] Note that $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ are sampled separately. There's no assumption that they are related to each other at all. Also, it's worth noting that a Dirichlet distribution *is* a Beta distribution if the dimension $V = 2$. Dirichlet generalizes Beta in the same way that multinomial generalizes binomial. Now you see why we took the trouble to represent $\gamma_{\pi 1}$ and $\gamma_{\pi 0}$ as a 2-dimensional vector $\boldsymbol{\gamma_\pi}$.

### 2.3 State space and initialization

Following Pedersen [17, 18], we're going to describe the Gibbs sampler in a completely unsupervised setting where no labels at all are provided as training data. We'll then briefly explain how to take advantage of labeled data.

**State space.** Recall that the job of a Gibbs sampler is to walk through an $k$-dimensional state space defined by the random variables $\langle Z_1, Z_2, \ldots Z_k \rangle$ in the model. Every point in that walk is a collection $\langle z_1, z_2, \ldots z_k \rangle$ of values for those variables.

In the Naïve Bayes model we've just described, here are the variables that define the state space.

- one scalar-valued variable $\pi$
- two vector-valued variables, $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$
- binary label variables $\mathbf{L}$, one for each of the $N$ documents

We also have one vector variable $\mathbf{W}_j$ for each of the $N$ documents, but these are observed variables, i.e. their values are already known (and which is why $W_{jk}$ is shaded in Figure 4).

**Initialization.** The initialization of our sampler is going to be very easy. Pick a value $\pi$ by sampling from the Beta$(\gamma_{\pi 1}, \gamma_{\pi 0})$ distribution. Then, for each $j$, flip a coin with success probability $\pi$, and assign label $\mathrm{L}_j^{(0)}$ — that is, the label of document $j$ at the $0^{th}$ iteration – based on the outcome of the coin flip. Similarly, you also need to initialize $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ by sampling from Dirichlet$(\boldsymbol{\gamma}_\theta)$.

### 2.4 Deriving the Joint Distribution

Recall that for each iteration $t = 1 \ldots T$ of sampling, we update every variable defining the state space by sampling from its conditional distribution given the other variables, as described in equation (14).

Here's how we're going to proceed:

- We will define the joint distribution of all the variables, corresponding to the numerator in (14).
- We simplify our expression for the joint distribution.
- We use our final expression of the joint distribution to define how to sample from the conditional distribution in (14).
- We give the final form of the sampler as pseudocode.

#### 2.4.1 Expressing and simplifying the joint distribution

According to our model, the joint distribution for the entire document collection is $P(\mathbb{C}, \mathbf{L}, \pi, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \gamma_{\pi 1}, \gamma_{\pi 0}, \boldsymbol{\gamma}_\theta)$. The semicolon indicates that the values to its right are parameters for this joint distribution. Another way to say this is that the variables to the left of the semicolon are conditioned on the hyperparameters given to the right of the semicolon. Using the model's generative story, and, crucially, the independence assumptions that are a part of that story, the joint distribution can decomposed into a product of several factors:[14]

$$P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) P(\mathbf{L}|\pi) P(\boldsymbol{\theta}_0|\boldsymbol{\gamma}_\theta) P(\boldsymbol{\theta}_1|\boldsymbol{\gamma}_\theta) P(\mathbb{C}_0|\boldsymbol{\theta}_0, \mathbf{L}) P(\mathbb{C}_1|\boldsymbol{\theta}_1, \mathbf{L})$$

Let's look at each of these in turn.

[14] Note that we can also obtain the products of the joint distribution directly from our graphical model, Figure 4 by multiplying together each latent variable conditioned on its parents.

- $P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0})$. The first factor is the probability of choosing this particular value of $\pi$ given that $\gamma_{\pi 1}$ and $\gamma_{\pi 0}$ are being used as the hyperparameters of the Beta distribution. By definition of the Beta distribution, that probability is:

$$P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) = \frac{\Gamma(\gamma_{\pi 1} + \gamma_{\pi 0})}{\Gamma(\gamma_{\pi 1})\Gamma(\gamma_{\pi 0})} \pi^{\gamma_{\pi 1}-1}(1-\pi)^{\gamma_{\pi 0}-1} \tag{17}$$

  And because

$$\frac{\Gamma(\gamma_{\pi 1} + \gamma_{\pi 0})}{\Gamma(\gamma_{\pi 1})\Gamma(\gamma_{\pi 0})}$$

  is a constant that doesn't depend on $\pi$, we can rewrite this as:

$$P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) = c\, \pi^{\gamma_{\pi 1}-1}(1-\pi)^{\gamma_{\pi 0}-1}. \tag{18}$$

  The constant $c$ is a normalizing constant that makes sure $P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0})$ sums to 1 over all $\pi$. $\Gamma(x)$ is the gamma function, a continuous-valued generalization of the factorial function. We could also express (18) as

$$P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) \propto \pi^{\gamma_{\pi 1}-1}(1-\pi)^{\gamma_{\pi 0}-1}. \tag{19}$$

- $P(\mathbf{L}|\pi)$. The second factor is the probability of obtaining this specific sequence $\mathbf{L}$ of $N$ binary labels, given that the probability of choosing label $= 1$ is $\pi$. That's

$$\begin{aligned} P(\mathbf{L}|\pi) &= \prod_{n=1}^{N} \pi^{\mathrm{L}_n}(1-\pi)^{(1-\mathrm{L}_n)} && (20) \\ &= \pi^{C_1}(1-\pi)^{C_0} && (21) \end{aligned}$$

  Recall from Figure 3 that $C_0$ and $C_1$ are nothing more than the number of documents labeled 1 and 0 respectively.[15]

- $P(\boldsymbol{\theta}_0|\boldsymbol{\gamma}_\theta)$ and $P(\boldsymbol{\theta}_1|\boldsymbol{\gamma}_\theta)$. The third factors are the probability of having sampled these particular choices of word distributions, given that $\boldsymbol{\gamma}_\theta$ was used as the hyperparameter of the Dirichlet distribution.

  Since the $\boldsymbol{\theta}$ distributions are independent of each other, let's make the notation a bit easier to read here and consider each of them in isolation, allowing us to momentarily elide the subscript saying which one is which. By the definition of the Dirichlet distribution, the probability of each word distribution is

$$\begin{aligned} P(\boldsymbol{\theta}|\boldsymbol{\gamma}_\theta) &= \frac{\Gamma(\sum_{i=1}^{V}\gamma_{\theta i})}{\prod_{i=1}^{V}\Gamma(\gamma_{\theta i})} \prod_{i=1}^{V} \theta_i^{\gamma_{\theta i}-1} && (22) \\ &= c' \prod_{i=1}^{V} \theta_i^{\gamma_{\theta i}-1} && (23) \\ &\propto \prod_{i=1}^{V} \theta_i^{\gamma_{\theta i}-1} && (24) \end{aligned}$$

  Recall that $\gamma_{\theta i}$ denotes the value of vector $\boldsymbol{\gamma}_\theta$'s $i^{th}$ dimension, and similarly, $\theta_i$ is the value for the $i^{th}$ dimension of vector $\boldsymbol{\theta}$, i.e. the probability assigned by this distribution to the $i^{th}$ word in the vocabulary. $c'$ is another normalization constant that we can discard by changing the equality to a proportionality.

[15] Of course we can represent these quantities given the variables we already have, but we define these in the interests of simplifying the equations somewhat.

- $P(\mathbb{C}_0|\boldsymbol{\theta}_0, \mathbf{L})$ and $P(\mathbb{C}_1|\boldsymbol{\theta}_1, \mathbf{L})$. These are the probabilities of generating the contents of the bags of words in each of the two document classes.

  Generating the bag of words $\mathbf{W}_n$ for document $n$ depends on that document's label, $\mathrm{L}_n$ and the word probability distribution associated with that label, $\boldsymbol{\theta}_{\mathrm{L}_n}$ (so $\boldsymbol{\theta}_{\mathrm{L}_n}$ is either $\boldsymbol{\theta}_0$ or $\boldsymbol{\theta}_1$). For notational simplicity, let's let $\boldsymbol{\theta} = \boldsymbol{\theta}_{\mathrm{L}_n}$:

$$P(\mathbf{W}_n|\mathbf{L}, \boldsymbol{\theta}_{\mathrm{L}_n}) = \prod_{i=1}^{V} \boldsymbol{\theta}_i^{\mathrm{W}_{ni}} \tag{25}$$

  Here $\theta_i$ is the probability of word $i$ in distribution $\boldsymbol{\theta}$, and the exponent $\mathrm{W}_{ni}$ is the frequency of word $i$ in $\mathbf{W}_n$.

  Now, since the documents are generated independently of each other, we can multiply the value in (25) for each of the documents in each class to get the combined probability of all the observed bags of words within a class:

$$P(\mathbb{C}_x|\mathbf{L}, \boldsymbol{\theta}_x) = \prod_{n \in \mathbb{C}_x} \prod_{i=1}^{V} \theta_{x,i}^{\mathrm{W}_{ni}} \tag{26}$$

$$= \prod_{i=1}^{V} \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i)} \tag{27}$$

  Where $\mathcal{N}_{\mathbb{C}_x}(i)$ gives the count of word $i$ in documents with class label $x$.

### 2.4.2 Choice of Priors and Simplifying the Joint Probability Expression

So why did we pick the Dirichlet distribution as our prior for $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ and the Beta distribution as our prior for $\pi$? Let's look at what happens in the process of simplifying the joint distribution and see what happens to our estimates of $\boldsymbol{\theta}$ (where this can be either $\boldsymbol{\theta}_0$ or $\boldsymbol{\theta}_1$) and $\pi$ once we observe some evidence (i.e. the words from a single document). Using (19) and (21) from above:

$$P(\pi|\mathbf{L}; \gamma_{\pi 1}, \gamma_{\pi 0}) = P(\mathbf{L}|\pi) P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) \tag{28}$$

$$\propto \left[\pi^{C_1}(1-\pi)^{C_0}\right] \left[\pi^{\gamma_{\pi 1}-1}(1-\pi)^{\gamma_{\pi 0}-1}\right] \tag{29}$$

$$\propto \pi^{C_1+\gamma_{\pi 1}-1}(1-\pi)^{C_0+\gamma_{\pi 0}-1} \tag{30}$$

Likewise, for $\boldsymbol{\theta}$ using (24) and (25) from above:

$$\begin{aligned} P(\boldsymbol{\theta}|\mathbf{W}_n; \boldsymbol{\gamma}_\theta) &= P(\mathbf{W}_n|\boldsymbol{\theta}) P(\boldsymbol{\theta}|\boldsymbol{\gamma}_\theta) \\ &\propto \prod_{i=1}^{V} \theta_i^{\mathrm{W}_{ni}} \prod_{i=1}^{V} \theta_i^{\gamma_{\theta i}-1} \\ &\propto \prod_{i=1}^{V} \theta_i^{\mathrm{W}_{ni}+\gamma_{\theta i}-1} \end{aligned} \tag{31}$$

If we use the words in all of the documents of a given class, then we have:

$$P(\boldsymbol{\theta}_x|\mathbb{C}_x; \boldsymbol{\gamma}_\theta) \propto \prod_{i=1}^{V} \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i}-1} \tag{32}$$

Notice that (30) is an unnormalized Beta distribution, with parameters $C_1 + \gamma_{\pi 1}$ and $C_0 + \gamma_{\pi 0}$, and (32) is an unnormalized Dirichlet distribution, with parameter vector $\langle \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta i} \rangle$ for $1 \leq i \leq V$. When the posterior probability distribution is of the same family as the likelihood probability distribution — that is, the same functional form, just with different arguments — it is said to be the *conjugate prior* of the posterior. The Beta distribution is the conjugate prior for binomial (and Bernoulli) distributions and the Dirichlet distribution is the conjugate prior for multinomial distributions. Also notice what role the hyperparameters played—they are added just like observed evidence. It is for this reason that the hyperparameters are sometimes referred to as *pseudocounts*.

Okay, so if we multiply together the individual factors from Section 2.4.1 as simplified above (in Section 2.4.2) and let $\boldsymbol{\mu} = \langle \gamma_{\pi 1}, \gamma_{\pi 0}, \boldsymbol{\gamma}_\theta \rangle$ we can express the full joint distribution as:

$$P(\mathbb{C}, \mathbf{L}, \pi, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu}) \quad \propto \quad \pi^{C_1 + \gamma_{\pi 1} - 1}(1-\pi)^{C_0 + \gamma_{\pi 0} - 1} \prod_{i=1}^{V} \theta_{0,i}^{\mathcal{N}_{\mathbb{C}_0}(i) + \gamma_{\theta i} - 1} \theta_{1,i}^{\mathcal{N}_{\mathbb{C}_1}(i) + \gamma_{\theta i} - 1} \tag{33}$$

### 2.4.3 Integrating out $\pi$

Having illustrated how to derive the joint distribution for a model, as in (33), it turns out that, for this particular model, we can make our lives a little bit simpler: we can reduce the effective number of parameters in the model by integrating our joint distribution with respect to $\pi$. This has the effect of taking all possible values of $\pi$ into account in our sampler, without representing it as a variable explicitly and having to sample it at every iteration. Intuitively, "integrating out" a variable is an application of precisely the same principle as computing the marginal probability for a discrete distribution. For example, if we have an expression for $P(a, b, c)$, we can compute $P(a, b)$ by summing over all possible values of $c$, i.e. $P(a, b) = \sum_c P(a, b, c)$. As a result, $c$ is "there" conceptually, in terms of our understanding of the model, but we don't need to deal with manipulating it explicitly as a parameter. With a continuous variable, the principle is the same, but we integrate over all possible values of the variable rather than summing.

So, we have

$$P(\mathbf{L}, \mathbb{C}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu}) = \int_\pi P(\mathbf{L}, \mathbb{C}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \pi; \boldsymbol{\mu}) \, \mathrm{d}\pi \tag{34}$$

$$= \int_\pi P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) P(\mathbf{L}|\pi) P(\boldsymbol{\theta}_0|\boldsymbol{\gamma}_\theta) P(\boldsymbol{\theta}_1|\boldsymbol{\gamma}_\theta) P(\mathbb{C}_0|\boldsymbol{\theta}_0, \mathbf{L}) P(\mathbb{C}_1|\boldsymbol{\theta}_1, \mathbf{L}) \, \mathrm{d}\pi \tag{35}$$

$$= P(\boldsymbol{\theta}_0|\boldsymbol{\gamma}_\theta) P(\boldsymbol{\theta}_1|\boldsymbol{\gamma}_\theta) P(\mathbb{C}_0|\boldsymbol{\theta}_0, \mathbf{L}) P(\mathbb{C}_1|\boldsymbol{\theta}_1, \mathbf{L}) \int_\pi P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) P(\mathbf{L}|\pi) \, \mathrm{d}\pi \tag{36}$$

At this point let's focus our attention on the integrand only and substitute the true distributions from (17) and (21).

$$\int_\pi P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) P(\mathbf{L}|\pi) \, \mathrm{d}\pi \quad = \quad \int_\pi \frac{\Gamma(\gamma_{\pi 1} + \gamma_{\pi 0})}{\Gamma(\gamma_{\pi 1})\Gamma(\gamma_{\pi 0})} \pi^{\gamma_{\pi 1} - 1}(1-\pi)^{\gamma_{\pi 0} - 1} \pi^{C_1}(1-\pi)^{C_0} \, \mathrm{d}\pi \tag{37}$$

$$= \quad \frac{\Gamma(\gamma_{\pi 1} + \gamma_{\pi 0})}{\Gamma(\gamma_{\pi 1})\Gamma(\gamma_{\pi 0})} \int_\pi \pi^{C_1 + \gamma_{\pi 1} - 1}(1-\pi)^{C_0 + \gamma_{\pi 0} - 1} \, \mathrm{d}\pi \tag{38}$$

Here's where our use of conjugate priors pays off. Notice that the integrand in (38) is a Beta distribution with parameters $C_1 + \gamma_{\pi 1}$ and $C_0 + \gamma_{\pi 0}$. This means that the value of the integral is just the normalizing constant for that distribution, which is easy to look up (e.g. in the entry for the Beta distribution in Wikipedia): the normalizing constant for distribution $\mathrm{Beta}(C_1 + \gamma_{\pi 1}, C_0 + \gamma_{\pi 0})$ is

$$\frac{\Gamma(C_1 + \gamma_{\pi 1})\Gamma(C_0 + \gamma_{\pi 0})}{\Gamma(C_0 + C_1 + \gamma_{\pi 1} + \gamma_{\pi 0})}$$

Making that substitution in (38), and also substituting $N = C_0 + C_1$, we arrive at

$$\int_{\pi} P(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) P(\mathbf{L}|\pi)\ \mathrm{d}\pi = \frac{\Gamma(\gamma_{\pi 1} + \gamma_{\pi 0})}{\Gamma(\gamma_{\pi 1})\Gamma(\gamma_{\pi 0})} \frac{\Gamma(C_1 + \gamma_{\pi 1})\Gamma(C_0 + \gamma_{\pi 0})}{\Gamma(N + \gamma_{\pi 1} + \gamma_{\pi 0})} \tag{39}$$

Substituting (39) and the definitions of the probability distributions from Section 2.4.1 back into (36) gives us

$$P(\mathbf{L}, \mathbb{C}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu}) \propto \frac{\Gamma(\gamma_{\pi 1} + \gamma_{\pi 0})}{\Gamma(\gamma_{\pi 1})\Gamma(\gamma_{\pi 0})} \frac{\Gamma(C_1 + \gamma_{\pi 1})\Gamma(C_0 + \gamma_{\pi 0})}{\Gamma(N + \gamma_{\pi 1} + \gamma_{\pi 0})} \prod_{i=1}^{V} \theta_{0,i}^{\mathcal{N}_{\mathbb{C}_0}(i) + \gamma_{\theta i} - 1} \theta_{1,i}^{\mathcal{N}_{\mathbb{C}_1}(i) + \gamma_{\theta i} - 1} \tag{40}$$

Okay, so it would be reasonable at this point to ask, "I thought the point of integrating out $\pi$ was to simplify things, so why did we just 'simplify' the joint expression by adding in a bunch of these Gamma functions everywhere?" Good question–hold that thought until we derive the sampler.

### 2.5 Building the Gibbs Sampler

The definition of a Gibbs sampler specifies that in each iteration we assign a new value to variable $Z_i$ by sampling from the conditional distribution

$$P(Z_i | z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \ldots, z_r^{(t)}).$$

So, for example, to assign the value of $\mathrm{L}_1^{(t+1)}$, we need to compute this conditional distribution:[16]

$$P(\mathrm{L}_1 | \mathrm{L}_2^{(t)}, \ldots, \mathrm{L}_N^{(t)}, \mathbb{C}, \boldsymbol{\theta}_0^{(t)}, \boldsymbol{\theta}_1^{(t)}; \boldsymbol{\mu}),$$

To assign the value of $\mathrm{L}_2^{(t+1)}$, we need to compute

$$P(\mathrm{L}_2 | \mathrm{L}_1^{(t+1)}, \mathrm{L}_3^{(t)}, \ldots, \mathrm{L}_N^{(t)}, \mathbb{C}, \boldsymbol{\theta}_0^{(t)}, \boldsymbol{\theta}_1^{(t)}; \boldsymbol{\mu}),$$

and so forth for $\mathrm{L}_3^{(t+1)}$ through $\mathrm{L}_N^{(t+1)}$. To assign the value of $\boldsymbol{\theta}_0$ we need to compute
Similarly, to assign the value of $\boldsymbol{\theta}_0$ we need to sample from the conditional distribution

$$P(\boldsymbol{\theta}_0^{(t+1)} | \mathrm{L}_1^{(t+1)}, \mathrm{L}_2^{(t+1)}, \ldots, \mathrm{L}_N^{(t+1)}, \mathbb{C}, \boldsymbol{\theta}_1^{(t)}; \boldsymbol{\mu}),$$

and, for $\boldsymbol{\theta}_1$,

$$P(\boldsymbol{\theta}_1^{(t+1)} | \mathrm{L}_1^{(t+1)}, \mathrm{L}_2^{(t+1)}, \ldots, \mathrm{L}_N^{(t+1)}, \mathbb{C}, \boldsymbol{\theta}_0^{(t+1)}; \boldsymbol{\mu}),$$

Intuitively, at the start of an iteration $t$, we have a collection of all our current information at this point in the sampling process. That information includes the word count for each document, the number of documents labeled 0, the number of documents labeled 1, the word counts for all documents labeled 0, the word counts for all documents labeled 1, the current label for each document, the current values of $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$, etc. When we want to sample the new label for document $j$, we temporarily remove *all* information (i.e. word counts and label information) about this document from that collection of information. Then we look at the conditional probability that $\mathrm{L}_j = 0$ given all the remaining information, and the conditional probability that $\mathrm{L}_j = 1$ given the same information, and we sample the new label $\mathrm{L}_j^{(t+1)}$ by choosing randomly according to the relative weight of those two conditional probabilities. Sampling to get the new values $\boldsymbol{\theta}_0^{(t+1)}$ and $\boldsymbol{\theta}_1^{(t+1)}$ operates according to the same principal.

[16] There's no superscript on the bags of words $\mathbb{C}$ because they're fully observed and don't change from iteration to iteration.

### 2.5.1 Sampling for Document Labels

Okay, so how do we actually do the sampling? We're almost there. As the final step in our journey, we show how to select all of the new document labels $\mathrm{L}_j$ and the new distributions $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ during each iteration of the sampler. By definition of conditional probability,

$$P(\mathrm{L}_j|\mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu}) = \frac{P(\mathrm{L}_j, \mathbf{W}_j, \mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu})}{P(\mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu})} \tag{41}$$

$$= \frac{P(\mathbf{L}, \mathbb{C}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu})}{P(\mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu})} \tag{42}$$

where $\mathbf{L}^{(-j)}$ are all the document labels except $\mathrm{L}_j$, and $\mathbb{C}^{(-j)}$ is the set of all documents except $\mathbf{W}_j$. The distribution is over two possible outcomes, $\mathrm{L}_j = 0$ and $\mathrm{L}_j = 1$.

Notice that the numerator is just the complete joint probability described in (40). In the denominator, we have the same expression, minus all the information about document $\mathbf{W}_j$. Therefore we can work out what (42) should look like by considering the three factors in (40), one at a time. For each factor, we will remind ourselves of what it looks like in the numerator (which includes $\mathbf{W}_j$), and work out what it should look like in the denominator (excluding $\mathbf{W}_j$), for each of the two outcomes.

The first factor in (40),

$$\frac{\Gamma(\gamma_{\pi 1} + \gamma_{\pi 0})}{\Gamma(\gamma_{\pi 1})\Gamma(\gamma_{\pi 0})}, \tag{43}$$

is very easy. It depends only on the hyperparameters, so removing information about $\mathbf{W}_j$ has no impact on its value. Since it will be the same in both the numerator and denominator of (42), it will cancel out. Excellent! Two factors to go.

Let's look at the second factor of (40). Taking all documents into account including $\mathbf{W}_j$, this is

$$\frac{\Gamma(C_1 + \gamma_{\pi 1})\Gamma(C_0 + \gamma_{\pi 0})}{\Gamma(N + \gamma_{\pi 1} + \gamma_{\pi 0})}. \tag{44}$$

Now, in order to compute the denominator of (42), we remove document $\mathbf{W}_j$ from consideration. How does that change things? It depends on what $\mathrm{L}_j$ was during the previous iteration. Whether $\mathrm{L}_j = 0$ or $\mathrm{L}_j = 1$ during the previous iteration, the corpus size is effectively reduced from $N$ to $N - 1$, and the size of one of the document classes is smaller by one compared to its value in the numerator. If $\mathrm{L}_j = 0$ when we remove it, then we will have $C_0^{(-j)} = C_0 - 1$ and $C_1^{(-j)} = C_1$. If $\mathrm{L}_j = 1$ during the previous iteration, then we will have $C_0 = C_0^{(-j)}$ and $C_1^{(-j)} = C_1 - 1$. In each case, removing $\mathbf{W}_j$ only changes the information we know about one class, which means that of the two terms in the numerator of this factor, one of them is going to be the same in the numerator and the denominator of (42). That's going to cause the terms from one of the classes (the one $\mathbf{W}_j$ did not belong to after the previous iteration) to cancel out.

Let $x \in \{0, 1\}$ be the outcome we're considering, i.e. the one for which $C_x^{(-j)} = C_x - 1$.[17] If we use $x$ and reorganize the expression a bit, the second factor of (42) can be rewritten from

$$\frac{\frac{\Gamma(C_1+\gamma_{\pi 1})\Gamma(C_0+\gamma_{\pi 0})}{\Gamma(N+\gamma_{\pi 1}+\gamma_{\pi 0})}}{\frac{\Gamma(C_0^{(-j)}+\gamma_{\pi 0})\Gamma(C_1^{(-j)}+\gamma_{\pi 1})}{\Gamma(N+\gamma_{\pi 1}+\gamma_{\pi 0}-1)}}$$

to

$$\frac{\Gamma(C_x + \gamma_{\pi x})\Gamma(N + \gamma_{\pi 1} + \gamma_{\pi 0} - 1)}{\Gamma(N + \gamma_{\pi 1} + \gamma_{\pi 0})\Gamma(C_x + \gamma_{\pi x} - 1)} \tag{45}$$

[17] Crucially, note that $x$ here is *not* $\mathrm{L}_j^{(t)}$, the label of document $j$ at the previous iteration. We are pulling document $j$ out of the collection, effectively obviating our knowledge of its current label, and then constructing a distribution over the two possibilities, $\mathrm{L}_j = 0$ and $\mathrm{L}_j = 1$. So we need for our expression to take $x$ as a parameter, allowing us to build a probability distribution by asking "what if $x = 0$?" and "what if $x = 1$?".

Using the fact that $\Gamma(a+1) = a\Gamma(a)$ for all $a$, we can simplify further to get

$$\frac{C_x + \gamma_{\pi x} - 1}{N + \gamma_{\pi 1} + \gamma_{\pi 0} - 1} \tag{46}$$

Look, no more pesky Gammas!

Finally, the third factor in (40) is

$$\prod_{i=1}^{V} \theta_{0,i}^{\mathcal{N}_{\mathbb{C}_0}(i)+\gamma_{\theta i}-1} \theta_{1,i}^{\mathcal{N}_{\mathbb{C}_1}(i)+\gamma_{\theta i}-1}. \tag{47}$$

When we look at what changes when we remove $\mathbf{W}_j$ from consideration, in order to write the corresponding expression in the denominator of (42), we see that it will behave in the same way as the second factor. One of the classes remains unchanged, so one of the terms, either the $\boldsymbol{\theta}_0$ or $\boldsymbol{\theta}_1$, will cancel out when we do the division. If, similar to above, we let $x$ be the class for which $C_x^{(-j)} = C_x - 1$, then we can again capture both cases in one expression:

$$\prod_{i=1}^{V} \frac{\theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i}-1}}{\theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x^{(-j)}}(i)+\gamma_{\theta i}-1}} = \prod_{i=1}^{V} \theta_{x,i}^{\mathrm{W}_{ji}} \tag{48}$$

With that, we have finished working through the three factors in (40), which means we have worked out how to express the numerator and denominator of (42), factor by factor. Recombining the factors, we get the following expression for the conditional distribution in (42): for $x \in \{0, 1\}$,

$$\Pr(\mathrm{L}_j = x | \mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1; \boldsymbol{\mu}) = \frac{C_x + \gamma_{\pi x} - 1}{N + \gamma_{\pi 1} + \gamma_{\pi 0} - 1} \prod_{i=1}^{V} \theta_{x,i}^{\mathrm{W}_{ji}} \tag{49}$$

Let's take a step back and take a look at what this equation is telling us about how a label is chosen. Its first factor gives us an indication of how likely it is that $\mathrm{L}_j = x$ considering only the distribution of the other labels. So, for example, if the corpus had more class 0 documents than class 1 documents, this factor would tend to push the label toward class 0. Its second factor is like a word distribution "fitting room." We get an indication of how well the words in $\mathbf{W}_j$ "fit" with each of the two distributions. If, for example, the words from $\mathbf{W}_j$ "fit" better with distribution $\boldsymbol{\theta}_0$ (by having a larger value for the document probability using $\boldsymbol{\theta}_0$) then this factor will push the label toward class 0 as well.

So (finally!) here's the actual procedure to sample from the conditional distribution in (42):

1. Let value0 = expression (49) with $x = 0$
2. Let value1 = expression (49) with $x = 1$
3. Let the distribution be $\langle \frac{value0}{value0+value1}, \frac{value1}{value0+value1} \rangle$
4. Select the value of $\mathrm{L}_j^{(t+1)}$ as the result of a Bernoulli trial (weighted coin flip) according to this distribution.

### 2.5.2 Sampling for $\boldsymbol{\theta}$

We'll follow a similar procedure to determine how to sample for new values of $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$. Since the estimation of the two distributions is independent of one another, we're going to omit the subscripts on $\boldsymbol{\theta}$ to make the notation a bit easier to digest. Just like above, we'll need to derive an expression for the probability of $\boldsymbol{\theta}$ given all other variables, but our work is a bit simpler in this case. Observe,

$$P(\boldsymbol{\theta} | \mathbb{C}, \mathbf{L}; \boldsymbol{\mu}) \propto P(\mathbb{C}, \mathbf{L} | \boldsymbol{\theta}) P(\boldsymbol{\theta} | \boldsymbol{\mu}) \tag{50}$$

Furthermore, recall that, since we used conjugate priors, this posterior, like the prior, works out to be a Dirichlet distribution. We actually derived the full expression in Section 2.4.2, but we don't need the full expression here. All we need to do to sample a new distribution is to make another draw from a Dirichlet distribution, but this time with parameters $\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta i}$ for each $i$ in $V$. For notational convenience, let's define the $V$ dimensional vector $\mathbf{t}$ such that each $t_i = \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta i}$, where $x$ is again either 0 or 1 depending on which $\boldsymbol{\theta}$ we are resampling. We then sample a new $\boldsymbol{\theta}$ as:

$$\boldsymbol{\theta} \sim \text{Dirichlet}(\mathbf{t}) \tag{51}$$

How do you actually implement sampling from a new Dirichlet distribution? To sample a random vector $\mathbf{a} = \langle a_1, \ldots, a_V \rangle$ from the $V$-dimensional Dirichlet distribution with parameters $\langle \alpha_1, \ldots, \alpha_V \rangle$, one fast way is to draw $V$ independent samples $y_1, \ldots, y_V$ from gamma distributions, each with density

$$\text{Gamma}(\alpha_i, 1) \;=\; \frac{y_i^{\alpha_i - 1}\, e^{-y_i}}{\Gamma(\alpha_i)}, \tag{52}$$

and then set $a_i = y_i / \sum_{j=1}^{V} y_j$ (i.e., just normalize each of the gamma distribution draws).[18]

### 2.5.3 Taking advantage of documents with labels

Using labeled documents is relatively painless: just don't sample $\mathrm{L}_j$ for those documents! Always keep $\mathrm{L}_j$ equal to the observed label. The documents will effectively serve as "ground truth" evidence for the distributions that created them. Since we never sample for their labels, they will *always* contribute to the counts in (49) and (51) and will never be subtracted out.

### 2.5.4 Putting it all together

**Initialization.** Define the priors as in Section 2.2 and initialize them as described in Section 2.3.

---

**for** $t := 1$ to $T$ **do**
  **for** $j := 1$ to $N$ **do**
    **if** $j$ is not a training document **then**
      Subtract $j$'s word counts from the total word counts of whatever class it's currently a member of
      Subtract 1 from the count of documents with label $\mathrm{L}_j$
      Assign a new label $\mathrm{L}_j^{(t+1)}$ to document $j$ as described at the end of Section 2.5.1
      Add 1 to the count of documents with label $\mathrm{L}_j^{(t+1)}$
      Add $j$'s word counts to the total word counts for class $\mathrm{L}_j^{(t+1)}$
    **end if**
  **end for**
  $\mathbf{t_0}$ := vector of total word counts from class 0, including pseudocounts
  $\boldsymbol{\theta}_0 \sim \text{Dirichlet}(\mathbf{t_0})$, as described in Section 2.5.2
  $\mathbf{t_1}$ := vector of total word counts from class 1, including pseudocounts
  $\boldsymbol{\theta}_1 \sim \text{Dirichlet}(\mathbf{t_1})$, as described in Section 2.5.2
**end for**

---

**Sampling iterations.** Notice that as soon as a new label for $\mathrm{L}_j$ is assigned, this changes the counts that will affect the labeling of the subsequent documents. This is, in fact, the whole principle behind a Gibbs sampler!

That concludes the discussion of how sampling is done. We'll see how to get from the output of the sampler to estimated values for the variables in Section 3.

[18] For details, see http://en.wikipedia.org/wiki/Dirichlet_distribution (version of April 12, 2010).

### 2.6 Optional: A Note on Integrating out Continuous Parameters

At this point you might be asking yourself why we were able to integrate out the continuous parameter $\pi$ from our model, but did not do something similar with the two word distributions $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$. The idea of doing this even *looks* promising, but there's a subtle problem that will get us into trouble and end up leaving us with an expression filled with $\Gamma$ functions that will not cancel out. Let us go through the derivations and see where it leads us. If you follow this piece of the discussion, then you *really* understand the details![19]

Our goal here would be to obtain the probability that a document was generated from the same distribution that generated the words of a particular class of documents, $\mathbb{C}_x$. We would then use this as a replacement for the product in (48). We start first by calculating the probability of making a *single* word draw given the other words in the class, subtracting out the information about $\mathbf{W}_j$. In the equations that follow there's an implicit $(-j)$ superscript on *all* of the counts. If we let $w_k$ denote the word at some position $k$ in $\mathbf{W}_j$ then,

$$
\begin{aligned}
\Pr(w_k = y|\mathbb{C}_x^{(-j)};\boldsymbol{\gamma}_\theta) &= \int_\Delta \Pr(w_k = y|\boldsymbol{\theta})P(\boldsymbol{\theta}|\mathbb{C}_x^{(-j)};\boldsymbol{\gamma}_\theta)d\boldsymbol{\theta} && (53)\\
&= \int_\Delta \theta_{x,y}\frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}\prod_{i=1}^V \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i}-1}d\boldsymbol{\theta} && (54)\\
&= \frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}\int_\Delta \theta_{x,y}\prod_{i=1}^V \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i}-1}d\boldsymbol{\theta} && (55)\\
&= \frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}\int_\Delta \theta_{x,y}^{\mathcal{N}_{\mathbb{C}_x}(y)+\gamma_{\theta y}}\prod_{i=1\wedge i\neq y}^V \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i}-1}d\boldsymbol{\theta} && (56)\\
&= \frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}\frac{\Gamma(\mathcal{N}_{\mathbb{C}_x}(y)+\gamma_{\theta y}+1)\prod_{i=1\wedge i\neq y}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})}{\Gamma(1+\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i})} && (57)\\
&= \frac{\mathcal{N}_{\mathbb{C}_x}(y)+\gamma_{\theta y}}{\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i)+\gamma_{\theta i}} && (58)
\end{aligned}
$$

The process we use is actually the same as the process used to integrate $\pi$, just in the multidimensional case. The set $\Delta$ is the probability simplex of $\boldsymbol{\theta}$, namely the set of all $\boldsymbol{\theta}$ such that $\sum_i \theta_i = 1$. We get to (54) by substitution from the formulas we derived in Section 2.4.2, then (55) by factoring out the normalization constant for the Dirichlet distribution, since it is constant with respect to $\boldsymbol{\theta}$. Note that the integrand of (56) is actually another Dirichlet distribution, so its integral is its normalization constant (same reasoning as before). We substitute this in to obtain (57). Using the property of $\Gamma$ that $\Gamma(x+1) = x\Gamma(x)$ for all $x$, we can again cancel all of the $\Gamma$ terms.

At this point, even though we have a simple and intuitive result for the probability of drawing a *single word* from a Dirichlet, we actually need the probability of drawing *all* words in $\mathbf{W}_j$ from the Dirichlet distribution. What we'd really like to do is assume that the words within a particular document are drawn from the same distribution, and just calculate the probability of $\mathbf{W}_j$ by multiplying (58) over all words in the vocabulary. But we cannot do that, since, without the values of $\boldsymbol{\theta}$ being known, we cannot make independent draws from a Dirichlet distribution since our draws have an effect on what our estimate of $\boldsymbol{\theta}$ is!

We can see this two different ways. First, instead of drawing one word in equation (53), do the derivation by drawing two words at a time.[20] You'll find that once you hit (57), you'll have an extra set of Gamma functions that won't cancel out nicely. The second way to see it is actually by looking at the plate diagram for the model, Figure 4. Each $\boldsymbol{\theta}$ effectively has $R_j$ arrows coming out of it for each document $j$ to individual

[19] The authors thank Wu Ke, who really understood the details, for pointing out our error in an earlier version of this document and providing the illustrating example we go through next.

[20] This is what Wu Ke did to demonstrate his point to us.

words, so every word within a document is in the Markov blanket [21] of the others; therefore we can't assume that the words are independent without a fixed $\boldsymbol{\theta}$. At issue here isn't just that there are multiple instances of words coming out of each theta, but crucially that those instances are sampled at the same time.

The lesson here is to be careful when you integrate out parameters. If you're doing a *single* draw from a multinomial, then integrating out a continuous parameter can make the sampler simpler, since you won't have to sample for it at every iteration. If, on the other hand, you do *multiple draws* from the same multinomial, integration (although possible) will result in an expression that involves Gamma functions. Calculating Gamma functions is undesirable since they are computationally expensive, so they can slow down a sampler significantly.

## 3 Producing values from the output of a Gibbs sampler

The initialization and sampling iterations in the Gibbs sampling algorithm will produce values for each of the variables, for iterations $t = 1, 2, \ldots, T$. In theory, the approximated value for any variable $Z_i$ can simply be obtained by calculating:

$$\frac{1}{T}\sum_{t=1}^{T} z_i^{(t)}, \tag{59}$$

as discussed in equation (12). However, expression (59) is not always used directly. There are several additional details to note that are a part of typical sampling practice.[22]

**Convergence and burn-in iterations.** Depending on the values chosen during the initialization step, it may take some time for the Gibbs sampler to reach a point where the points $\langle z_1^{(t)}, z_2^{(t)}, \ldots z_r^{(t)} \rangle$ are all coming from the stationary distribution of the Markov chain, which is an assumption of the approximation in (59). In order to avoid the estimates being contaminated by the values at iterations before that point, some practitioners generally discard the values at iterations $t < B$, which are referred to as the "burn-in" iterations, so that the average in (59) is taken only over iterations $B + 1$ through $T$.[23]

**Autocorrelation and lag.** The approximation in (59) assumes the samples for $Z_i$ are independent, but we know they're not, because they were produced by a process that conditions on the previous point in the chain to generate the next point. This is referred to as *autocorrelation* (sometimes *serial autocorrelation*), i.e. correlation between successive values in the data.[24] In order to avoid autocorrelation problems (so that the chain "mixes well"), many implementations of Gibbs sampling average only every $L^{th}$ value, where $L$ is referred to as the *lag*.[25] In this context, Jordan Boyd-Graber (personal communication) also recommends looking at Neal's [15] discussion of likelihood as a metric of convergence.

[21] The Markov blanket of a node in a graphical model consists of that node's parents, its children, and the coparents of its children. [16].

[22] Jason Eisner (personal communication) argues, with some support from the literature, that burn-in, lag, and multiple chains are in fact unnecessary and it is perfectly correct to do a single long sampling run and keep all samples. See [4, 5], MacKay ([13], end of section 29.9, page 381) and Koller and Friedman ([10], end of section 12.3.5.2, page 522).

[23] As far as we can tell, there is no principled way to choose the "right" value for $B$ in advance. There are a variety of ways to test for convergence, and to measure autocorrelation; see, e.g., Brian J. Smith, "boa: An R Package for MCMC Output Convergence Assessment and Posterior Inference", Journal of Statistical Software, November 2007, Volume 21, Issue 11, http://www.jstatsoft.org/ for practical discussion. However, from what we can tell, many people just choose a really big value for $T$, pick $B$ to be large also, and assume that their samples are coming from a chain that has converged.

[24] Lohninger [11] observes that "most inferential tests and modeling techniques fail if data is autocorrelated".

[25] Again, the choice of $L$ seems to be more a matter of art than science: people seem to look at plots of the autocorrelation for different values of $L$ and use a value for which the autocorrelation drops off quickly. The autocorrelation for variable $Z_i$ with lag $L$ is simply the correlation between the sequence $Z_i^{(t)}$ and the sequence $Z_i^{(t-L)}$. Which correlation function is used seems to vary.

**Multiple chains.** As is the case for many other stochastic algorithms (e.g. expectation maximization as used in the forward-backward algorithm for HMMs), people often try to avoid sensitivity to the starting point chosen at initialization time by doing multiple runs from different starting points. For Gibbs sampling and other Markov Chain Monte Carlo methods, these are referred to as "multiple chains".[26]

**Hyperparameter sampling.** Rather than simply picking hyperparameters, it is possible, and in fact often critical, to assign their values via sampling (Boyd-Graber, personal communication). See, e.g., Wallach et al. [20] and Escobar and West [3].

## 4 Conclusions

The main point of this document has been to take some of the mystery out of Gibbs sampling for computer scientists who want to get their hands dirty and try it out. Like any other technique, however, *caveat lector*: using a tool with only limited understanding of its theoretical foundations can produce undetected mistakes, misleading results, or frustration.

As a first step toward getting further up to speed on the relevant background, Ted Pedersen's [18] doctoral dissertation has a very nice discussion of parameter estimation in Chapter 4, including a detailed exposition of an EM algorithm for Naïve Bayes and his own derivation of a Naïve Bayes Gibbs sampler that highlights the relationship to EM. (He works through several iterations of each algorithm explicitly, which in our opinion merits a standing ovation.) The ideas introduced in Chapter 4 are applied in Pedersen and Bruce [17]; note that the brief description of the Gibbs sampler there makes an *awful* lot more sense after you've read Pedersen's dissertation chapter.

We also recommend Gregor Heinrich's [7] "Parameter estimation for text analysis." Heinrich presents fundamentals of Bayesian inference starting with a nice discussion of basics like maximum likelihood estimation (MLE) and maximum a posteriori (MAP) estimation, all with an eye toward dealing with text. (We followed his discussion closely above in Section 1.1.) Also, his is one of the few papers we've been able to find that actually provides pseudo-code for a Gibbs sampler. Heinrich discusses in detail Gibbs sampling for the widely discussed Latent Dirichlet Allocation (LDA) model, and his corresponding code is at http://www.arbylon.net/projects/LdaGibbsSampler.java.

For a textbook-style exposition, see Bishop [2]. The relevant pieces of the book are a little less standalone than we'd like (which makes sense for a course on machine learning, as opposed to just trying to dive straight into a specific topic); Chapter 11 (Sampling Methods) is most relevant, though you may also find yourself referring back to Chapter 8 (Graphical Models).

Those ready to dive into the topic of Markov Chain Monte Carlo in more depth might want to start with Andrieu et al. [1]. We and Andrieu et al. appear to differ somewhat on the semantics of the word "introduction," which is one of the reasons the document you're reading exists.

Finally, for people interested in the use of Gibbs sampling for *structured* models in NLP (e.g. parsing), the right place to start is undoubtedly Kevin Knight's excellent "Bayesian Inference with Tears: A tutorial workbook for natural language researchers" [9], after which you'll be equipped to look at Johnson, Griffiths, and Goldwater [8].[27] The leap from the models discussed here to those kinds of models actually turns out to be a lot less scary than it appears at first. The main thing to observe is that in the crucial sampling step (equation (14) of Section 1.2.3), the denominator is just the numerator without $z_i^{(t)}$, the variable whose new value you're choosing. So when you're sampling conditional distributions (e.g. Sections 2.5.1–2.5.4) in more complex models, the basic idea will be the same: you subtract out counts related to the variable you're interested in based on its current value, compute proportions based on the remaining counts, then

[26] Again, there seems to be as much art as science in whether to use multiple chains, how many, and how to combine them to get a single output. Chris Dyer (personal communication) reports that it is not uncommon simply to concatenate the chains together after removing the burn-in iterations.

[27] As an aside, our travels through the literature in writing this document led to an interesting early use of Gibbs sampling with CFGs: Grate et al. [6]. Johnson et al. had not come across this when they wrote their seminal paper introducing Gibbs sampling for PCFGs to the NLP community (and in fact a search on scholar.google.com turned up no citations in the NLP literature). Mark Johnson (personal communication) observes that the "local move" Gibbs sampler Grate et al. describe is specialized to a particular PCFG, and it's not clear how to generalize it to arbitrary PCFGs.

pick probabilistically based on the result, and finally add counts back in according to the probabilistic choice you just made.

**Acknowledgments**

The creation of this document has been supported in part by the National Science Foundation (award IIS-0838801), the GALE program of the Defense Advanced Research Projects Agency (Contract No. HR0011-06-2-001), and the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the Army Research Laboratory. All statements of fact, opinion, or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of NSF, DARPA, IARPA, the ODNI, the U.S. Government, the University of Maryland, nor of the Resnik or Hardisty families or any of their close relations, friends, or family pets.

The authors are grateful to Jordan Boyd-Graber, Bob Carpenter, Chris Dyer, Jason Eisner, John Goldsmith, Kevin Knight, Mark Johnson, Nitin Madnani, Neil Parikh, Sasa Petrovic, William Schuler, Prithvi Sen, Wu Ke and Matt Pico (so far!) for helpful comments and/or catching glitches in the manuscript. Extra thanks to Jordan Boyd-Graber for many helpful discussions and extra assistance with plate diagrams, and to Jay Resnik for his help debugging the LaTeX document.

**References**

[1] Andrieu, Freitas, Doucet, and Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[2] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, June 1995.

[4] C. Geyer. Burn-in is unnecessary, 2009. http://www.stat.umn.edu/∼charlie/mcmc/burn.html, Downloaded October 18, 2009.

[5] C. Geyer. One long run, 2009. http://www.stat.umn.edu/∼charlie/mcmc/one.html.

[6] L. Grate, M. Herbster, R. Hughey, D. Haussler, I. S. Mian, and H. Noller. Rna modeling using gibbs sampling and stochastic context free grammars. In *ISMB*, pages 138–146, 1994.

[7] G. Heinrich. Parameter estimation for text analysis. Technical Note Version 2.4, vsonix GmbH and University of Leipzig, August 2008. http://www.arbylon.net/publications/text-est.pdf.

[8] M. Johnson, T. Griffiths, and S. Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April 2007. Association for Computational Linguistics.

[9] K. Knight. Bayesian inference with tears: A tutorial workbook for natural language researchers, 2009. http://www.isi.edu/natural-language/people/bayes-with-tears.pdf.

[10] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[11] H. Lohninger. *Teach/Me Data Analysis*. Springer-Verlag, 1999. http://www.vias.org/tmdatanaleng/cc_corr_auto_1.html.

[12] A. Lopez. Statistical machine translation. *ACM Computing Surveys*, 40(3):1–49, August 2008.

[13] D. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[14] C. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[15] R. M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993. http://www.cs.toronto.edu/∼radford/ftp/review.pdf.

[16] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[17] T. Pedersen. Knowledge lean word sense disambiguation. In *AAAI/IAAI*, page 814, 1997.

[18] T. Pedersen. *Learning Probabilistic Models of Word Sense Disambiguation*. PhD thesis, Southern Methodist University, 1998. http://arxiv.org/abs/0707.3972.

[19] S. Theodoridis and K. Koutroumbas. *Pattern Recognition, 4th Ed.* Academic Press, 2009.

[20] H. Wallach, C. Sutton, and A. McCallum. Bayesian modeling of dependency trees using hierarchical Pitman-Yor priors. In *ICML Workshop on Prior Knowledge for Text and Language Processing*, 2008.