

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@umd.edu

Eric Hardisty

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

October 20, 2009

VERSION 0.3

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, and 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 two applications in detail. The first is the derivation of a Gibbs sampler for Naive Bayes models, which illustrates a simple case where the math works out very cleanly and it’s possible to “integrate out” the model’s continuous parameters to build a more efficient algorithm. The second application derives the Gibbs sampler for a model that is similar to Naive Bayes, but which adds an additional latent variable. Having gone through the two examples, we discuss some practical implementation issues. 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 [16], 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

¹This subsection is built around the very nice explication of Bayesian probability estimation by Heinrich [8].

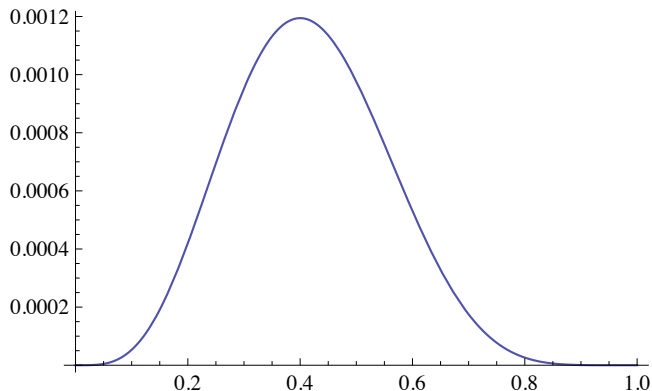


Figure 1: Probability of generating the coin-flip sequence HHHHTTTTTT, using different values for $\text{Pr}(\text{heads})$ on the x -axis. The value that maximizes the probability of the observed sequence, 0.4, is the maximum likelihood estimate (MLE).

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 *maximum 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.

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

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

In this case, the most natural model μ has just a single parameter, π , 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

$$\tilde{\pi}_{MLE} = \underset{\pi}{\operatorname{argmax}} \operatorname{Pr}(\mathcal{X}|\pi) \quad (2)$$

$$\operatorname{Pr}(y|\mathcal{X}) \approx \operatorname{Pr}(y|\tilde{\pi}_{MLE}) \quad (3)$$

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,

$$\begin{aligned} \tilde{\pi}_{MAP} &= \underset{\pi}{\operatorname{argmax}} \operatorname{Pr}(\pi|\mathcal{X}) \\ &= \underset{\pi}{\operatorname{argmax}} \frac{\operatorname{Pr}(\mathcal{X}|\pi) \operatorname{Pr}(\pi)}{\operatorname{Pr}(\mathcal{X})} \\ &= \underset{\pi}{\operatorname{argmax}} \operatorname{Pr}(\mathcal{X}|\pi) \operatorname{Pr}(\pi) \end{aligned} \quad (4)$$

$$\operatorname{Pr}(y|\mathcal{X}) \approx \operatorname{Pr}(y|\tilde{\pi}_{MAP}) \quad (5)$$

²Specifically, μ models each choice as a Bernoulli trial, and the probability of generating exactly this heads-tails sequence for a given π 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 [8], Section 2.1.

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 π to be in the form of a prior probability distribution $\Pr(\pi)$.³ So, 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 $\Pr(\pi) = 0.5$. 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, π could have *any* value between 0 and 1; mightn’t we get better estimates if we took the whole distribution $p(\pi|\mathcal{X})$ into account, rather than just a single estimated value for π ? If we do that, we’re making use of all the information about π that we can wring from the observed \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 π . 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:

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

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

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

where Bayes’s Rule defines

$$\Pr(\pi|\mathcal{X}) = \frac{\Pr(\mathcal{X}|\pi) \Pr(\pi)}{\Pr(\mathcal{X})} \tag{9}$$

$$= \frac{\Pr(\mathcal{X}|\pi) \Pr(\pi)}{\int_{\pi} \Pr(y|\pi) \Pr(\pi) d\pi}. \tag{10}$$

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

Equations (8) and (10) provide one compelling answer to the question we started with. Why should even discrete-minded computer scientists care about integrals? Because computing integrals makes it possible to compute expected values, and those can help us improve the parameter estimates in our models.⁵

³We 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.

⁴See <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>.

⁵Chris 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 [14]) 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” [10] 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. Here we'll briefly describe why Gibbs sampling makes things easier.

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 π (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

$$\frac{C}{S} \approx \frac{\pi(\frac{d}{2})^2}{d^2}. \quad (11)$$

Solving for π , 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 infinitesimally small points; similarly for 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 draw N points $z^{(0)}, z^{(1)}, z^{(2)}, \dots, z^{(N)}$ from the probability density $p(z)$, then

$$E_{p(z)}[f(z)] = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{t=1}^N f(z^{(t)}). \quad (12)$$

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

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

⁷Okay — 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?

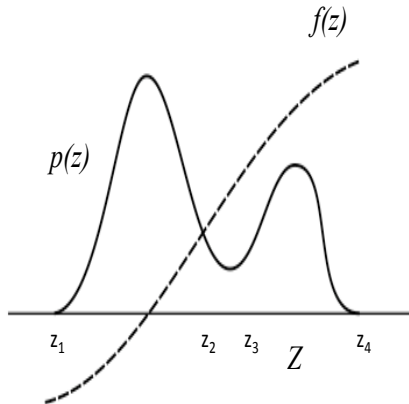


Figure 2: Example of computing expectation $E_{p(z)}[f(z)]$. (This figure was adapted from page 7 of the handouts for Chris Bishop’s presentation “NATO ASI: Learning Theory and Practice”, Leuven, July 2002, <http://research.microsoft.com/en-us/um/people/cmbishop/downloads/bishop-nato-2.pdf>)

Looking at equation (12), it’s clear that we can get an *approximate* value by sampling only a finite number of times, T :

$$E_{p(z)}[f(z)] \approx \frac{1}{T} \sum_{t=1}^T f(z^{(t)}). \quad (13)$$

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

Let $z^{(0)}$ = a random initial point

For $t = 1..T$,

$$z^{(t+1)} = g(z^{(t)})$$

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 $p_{\text{trans}}(z^{(t+1)}|z^{(0)}, z^{(1)}, \dots, z^{(t)})$.⁸

⁸We’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.

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,

$$p_{\text{trans}}(z^{(t+1)}|z^{(0)}, z^{(1)}, \dots, z^{(t)}) = p_{\text{trans}}(z^{(t+1)}|z^{(t)}). \quad (14)$$

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_{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_{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 $z = \langle z_1, \dots, z_k \rangle$, 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:

Pick values for $z^{(0)} = \langle z_1^{(0)}, \dots, z_k^{(0)} \rangle$. This is your initial state.

For $t = 1..T$,

For $i = 1..k$, pick value $z_i^{(t+1)}$ by sampling from the following distribution:

$$\Pr(Z_i|z_1^{(t+1)}, \dots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \dots, z_k^{(t)}) = \frac{\Pr(z_1^{(t+1)}, \dots, z_{i-1}^{(t+1)}, z_i^{(t)}, z_{i+1}^{(t)}, \dots, z_k^{(t)})}{\Pr(z_1^{(t+1)}, \dots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \dots, z_k^{(t)})}. \quad (15)$$

– You’ve just computed your new point $z^{(t+1)} = g(z^{(t)}) = \langle z_1^{(t+1)}, \dots, z_k^{(t+1)} \rangle$.

You can think of each dimension z_i as corresponding to a parameter or variable in your model. Using equation (15), 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 .
- 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 easier to obtain the values for 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.

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

¹⁰There 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.

Well, ok... but. 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 15) 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 — Naive 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 to integrate out continuous parameters (drastically simplifying what you’ll need to implement), and how to actually sample from conditional distributions per equation (15).

In Section 3 we go through a second example. Based on the Latent Sentence Perspective Model [12], this model is structured very similarly to the Naive Bayes model as we describe it in Section 2 (which predicts document classes based on words), but it adds a single latent variable (at the level of sentences). Once you’ve understood what’s going on here, you should be equipped to move on to more complex latent variable models such as Latent Dirichlet Allocation (LDA).¹¹

In Section 4 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 5 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 Naive Bayes Model

In this section we consider Naive Bayes models.¹² 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. This makes the discussion concrete, and uses the same problem setting we’ll also use in Section 3 when we talk about a slightly more complicated model. For ease of reference, we present our notation in Figure 3. Figure 4 describes the model as a “plate diagram”, to which we refer as the model is described.

2.1 Modeling How Documents are Generated

Given an unlabeled document \mathbf{W}_j , our goal is to pick the best label L_j , and hence its membership in either \mathbb{C}_0 or \mathbb{C}_1 . The usual treatment of these models is to equate “best” with “most probable”, and therefore our goal is to choose the label L_j that maximizes $\Pr(L_j|\mathbf{W}_j)$. Applying Bayes’s Rule,

$$\begin{aligned} \operatorname{argmax}_{L_j} \Pr(L_j|\mathbf{W}_j) &= \operatorname{argmax}_{L_j} \frac{\Pr(\mathbf{W}_j|L_j) \Pr(L_j)}{\Pr(\mathbf{W}_j)} \\ &\propto \operatorname{argmax}_{L_j} \Pr(\mathbf{W}_j|L_j) \Pr(L_j), \end{aligned}$$

where the denominator $\Pr(\mathbf{W}_j)$ is omitted (changing the equality to a proportionality) because it does not depend on L_j .

This application of Bayes’s rule — the “Bayes” part of “Naive Bayes” — makes it natural to think of the model in terms of a “generative story” that accounts for how documents are created. According to that

¹¹We don’t cover LDA, but we’d recommend going from here to Heinrich [8] to learn about it. See Section 5.

¹²This section owes a great deal to Prithvi Sen’s patient and gracious explanation of how to derive a Naive Bayes Gibbs sampler in which the continuous parameters have been integrated out.

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.
γ_{θ}	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.
W_{ji}	frequency of word i in document j .
\mathbf{L}	vector of document labels.
L_j	label for document j .
θ_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 .
$\mathcal{N}_{\mathbf{W}_j}(i)$	number of times word i occurs in \mathbf{W}_j
$\mathbb{C}^{(-j)}$	set of all documents <i>except</i> \mathbf{W}_j
$\mathbb{C}_x^{(-j)}$	set of all documents from class x <i>except</i> \mathbf{W}_j
$\mathbf{L}^{(-j)}$	vector of all document labels <i>except</i> L_j
$C_0^{(-j)} (C_1^{(-j)})$	number of documents labeled 0 (1) <i>except</i> for \mathbf{W}_j

Figure 3: Notation for the discussion of Naive Bayes.

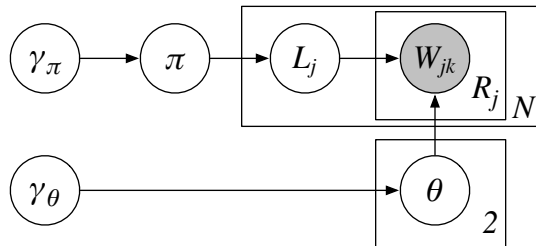


Figure 4: Naive Bayes plate diagram

story, we first pick the class L_j of the document; our model will assume that’s done by flipping a coin whose probability of heads is some value $\pi = \Pr(L_j = 1)$. We can express this a little more formally as

$$L_j \sim \text{Bernoulli}(\pi).$$

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 L_j of the document, so we’ll write them as θ_0 and θ_1 . Formally one would describe the creation of document j ’s bag of words as:

$$\mathbf{W}_j \sim \text{Multinomial}(\theta). \tag{16}$$

The assumption that the words are chosen independently is the reason we call the model “naive”.

Notice that logically speaking, it made sense to describe the model in terms of two separate probability distributions, θ_0 and θ_1 , each one being a simple unigram distribution. The notation in (16) doesn’t explicitly show that what happens in generating \mathbf{W}_j depends on whether L_j was 0 or 1. Unfortunately, that notational choice seems to be standard, even though it’s less transparent.¹³ We indicate the existence of two θ s by including the 2 in the lower rectangle of Figure 4, but many plate diagrams in the literature would not.

And that’s it: our “generative story” for the creation of a whole set of labeled documents $\langle \mathbf{W}_n, L_n \rangle$, according to the Naive 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 π come from? Our generative story is going to assume that before this whole process began, we also picked π randomly. Specifically we’ll assume that π 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 $\gamma_\pi = \langle \gamma_{\pi 1}, \gamma_{\pi 0} \rangle$. When $\gamma_{\pi 1} = \gamma_{\pi 0} = 1$, $\text{Beta}(\gamma_{\pi 1}, \gamma_{\pi 0})$ is just a uniform distribution, which means that any value for π is equally likely. For this reason we call $\text{Beta}(1, 1)$ an “uninformed prior”.

Similarly, where do θ_0 and θ_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 γ_θ be a V -dimensional vector where the value of every dimension equals 1. If θ_0 is sampled from $\text{Dirichlet}(\gamma_\theta)$, every probability distribution over words will be equally likely. Similarly, we’ll assume θ_1 is sampled from $\text{Dirichlet}(\gamma_\theta)$.¹⁴

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.

2.3 State space and initialization

Following Pedersen [18, 19], 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.

¹³The actual basis for removing the subscripts on the parameters θ 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 [20].

¹⁴Note that θ_0 and θ_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 γ_π .

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

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

- one scalar-valued variable π
- two vector-valued variables, θ_0 and θ_1
- binary label variables \mathbf{L} , one for each of the N documents

We also have one variable \mathbf{W}_j for each of the N documents, but these are observed variables, i.e. their values are already known.

Initialization. The initialization of our sampler is going to be very easy. Pick a value π by sampling from the $Beta(\gamma_{\pi 1}, \gamma_{\pi 0})$ distribution. Then, for each j , flip a coin with success probability π , and assign label $L_j^{(0)}$ — that is, the label of document j at the 0^{th} iteration — based on the outcome of the coin flip. You might think you also need to initialize θ_0 and θ_1 by sampling from a Dirichlet distribution, but it will turn out that you don't. In fact, when implementing this Gibbs sampler you'll never need to define those variables at all!

2.4 Sampling Iterations

Recall that for each iteration $t = 1 \dots T$ of sampling, we update every variable defining the state space by sampling from its conditional distribution given the other variables, as described in equation (15).

Here's how we're going to proceed:

- We will define the joint distribution of all the variables, corresponding to the numerator in (15).
- We simplify our expression for the joint distribution.
- We take that simplified expression and show how to simplify things still further by integrating out some of the variables. This will greatly reduce the amount of work the sampler needs to do (and the amount of work you need to do when implementing it). In fact, we'll see that the only variables we'll need to keep track of are the counts for the number of documents with each label, and the word frequencies for each document.
- We use our final expression of the joint distribution to define how to sample from the conditional distribution.
- 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 is $\Pr(\mathbf{C}, \mathbf{L}, \pi, \theta_0, \theta_1; \gamma_{\pi 1}, \gamma_{\pi 0}, \gamma_\theta)$.¹⁵ Using the model's generative story, and, crucially, the independence assumptions that are a part of that story, the joint distribution can be decomposed into a product of several factors:

$$\Pr(\pi | \gamma_{\pi 1}, \gamma_{\pi 0}) \times \Pr(\mathbf{L} | \pi) \times \Pr(\theta_0 | \gamma_\theta) \times \Pr(\theta_1 | \gamma_\theta) \times \Pr(\mathbf{C}_0 | \theta_0) \times \Pr(\mathbf{C}_1 | \theta_1)$$

Let's look at each of these in turn. (If you don't care to see the algebra worked out, skip to Section 2.5.)

¹⁵The 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.

- $\Pr(\pi|\gamma_{\pi 1}, \gamma_{\pi 0})$. The first factor is the probability of choosing this particular value of π 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:

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

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 π , we can rewrite this as:

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

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

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

- $\Pr(\mathbf{L}|\pi)$. 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 π . That's

$$\Pr(\mathbf{L}|\pi) = \prod_{n=1}^N \pi^{L_n} (1 - \pi)^{(1-L_n)} \quad (20)$$

$$= \pi^{C_1} (1 - \pi)^{C_0} \quad (21)$$

Recall from Figure 3 that C_0 and C_1 are nothing more than the number of documents labeled 1 and 0 respectively.¹⁶

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

Let's consider each of the word distributions in isolation, allowing us to momentarily elide the extra subscript. By the definition of the Dirichlet distribution the probability of each word distribution is

$$\Pr(\boldsymbol{\theta}|\gamma_\theta) = \frac{\Gamma(\sum_{i=1}^V \gamma_{\theta i})}{\prod_{i=1}^V \Gamma(\gamma_{\theta i})} \prod_{i=1}^V \theta_i^{\gamma_{\theta i}-1} \quad (22)$$

$$= c' \prod_{i=1}^V \theta_i^{\gamma_{\theta i}-1} \quad (23)$$

$$\propto \prod_{i=1}^V \theta_i^{\gamma_{\theta i}-1} \quad (24)$$

Recall that $\gamma_{\theta i}$ denotes the value of vector γ_θ 's i^{th} dimension, and similarly, θ_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.

¹⁶Of course, we could also represent $|C_0|$ as $N - C_1$, but we chose the former to make the notation more compact.

- $\Pr(\mathbb{C}_0|\boldsymbol{\theta}_0)$ and $\Pr(\mathbb{C}_1|\boldsymbol{\theta}_1)$. 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, L_n and the word probability distribution associated with that label, $\boldsymbol{\theta}_{L_n}$ (so $\boldsymbol{\theta}_{L_n}$ is either $\boldsymbol{\theta}_0$ or $\boldsymbol{\theta}_1$):

$$\Pr(\mathbf{W}_n|L_n, \boldsymbol{\theta}_{L_n}) = \prod_{i=1}^V \theta_{L_n, i}^{\mathcal{N}_{\mathbf{W}_n}(i)} \quad (25)$$

Here θ_i is the probability of word i in distribution $\boldsymbol{\theta}$, i.e. $\Pr(W_i|\boldsymbol{\theta})$, and the exponent $\mathcal{N}_{\mathbf{W}_n}(i)$ 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:

$$\Pr(\mathbb{C}_x|\mathbf{L}, \boldsymbol{\theta}_x) = \prod_{n=1}^{C_x} \prod_{i=1}^V \theta_{x, i}^{\mathcal{N}_{\mathbf{W}_n}(i)} \quad (26)$$

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

2.4.2 Choice of Priors

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 π ? Let's look at what happens to our estimate of $\boldsymbol{\theta}$ and π once we observe some evidence (the words from a single document). Using (24) from above:

$$\Pr(\pi|\mathbf{L}; \gamma_{\pi 1}, \gamma_{\pi 0}) \propto \Pr(\mathbf{L}|\pi) \Pr(\pi|\gamma_{\pi 1}, \gamma_{\pi 0}) \quad (28)$$

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

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

Likewise, for $\boldsymbol{\theta}$:

$$\begin{aligned} \Pr(\boldsymbol{\theta}|\mathbf{W}_n; \boldsymbol{\gamma}_\theta) &\propto \Pr(\mathbf{W}_n|\boldsymbol{\theta}) \Pr(\boldsymbol{\theta}|\boldsymbol{\gamma}_\theta) \\ &\propto \prod_{i=1}^V \theta_i^{\mathcal{N}_{\mathbf{W}_n}(i)} \prod_{i=1}^V \theta_i^{\gamma_{\theta i}-1} \\ &= \prod_{i=1}^V \theta_i^{\mathcal{N}_{\mathbf{W}_n}(i)+\gamma_{\theta i}-1} \end{aligned} \quad (31)$$

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

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

Notice that (30) is another Beta distribution, with parameters $C_1 + \gamma_{\pi 1}$ and $C_0 + \gamma_{\pi 0}$, and (32) is another Dirichlet distribution, with parameters $\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta i}$. 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. To change the proportionality to an equality, we need to normalize using each distribution's respective normalizing constant, leaving us with:

$$\Pr(\pi|\mathbf{L}; \gamma_{\pi_1}, \gamma_{\pi_0}) = \frac{\Gamma(N + \gamma_{\pi_1} + \gamma_{\pi_0})}{\Gamma(C_1 + \gamma_{\pi_1})\Gamma(C_0 + \gamma_{\pi_0})} \pi^{C_1 + \gamma_{\pi_1} - 1} (1 - \pi)^{C_0 + \gamma_{\pi_0} - 1} \quad (33)$$

$$\Pr(\boldsymbol{\theta}_x|\mathbb{C}_x; \boldsymbol{\gamma}_\theta) = \frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})} \prod_{i=1}^V \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i} - 1} \quad (34)$$

Note that (33) is the same as the normalized product of (19) and (21), and that (34) is the same as the normalized product of (24) and (27).

Instead of trying to directly calculate π , $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ to use for future estimates (as one would with expectation maximization), we can derive an equation that gives the probability of generating another document of a particular class, given the other documents of that class. In other words, we'd like to calculate the probability that label $L_j = 1$ is generated given the other labels as evidence, and we want to calculate the probability of generating a bag of words given the sets \mathbb{C}_0 and \mathbb{C}_1 as evidence. This actually simplifies the sampler since it makes the continuous-valued parameters disappear. (If you don't care about the details, skip to Section 2.5.3.)

2.5 Integrating out continuous parameters

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

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

So, for example, to assign the value of $\pi^{(t+1)}$, we would need to compute

$$\Pr(\pi|L_1^{(t)}, \dots, L_N^{(t)}, \mathbb{C}, \boldsymbol{\theta}_0^{(t)}, \boldsymbol{\theta}_1^{(t)}; \gamma_{\pi_1}, \gamma_{\pi_0}, \boldsymbol{\gamma}_\theta),$$

to assign the value of $L_1^{(t+1)}$, we need to compute

$$\Pr(L_1|\pi^{(t+1)}, L_2^{(t)}, \dots, L_N^{(t)}, \mathbb{C}, \boldsymbol{\theta}_0^{(t)}, \boldsymbol{\theta}_1^{(t)}; \gamma_{\pi_1}, \gamma_{\pi_0}, \boldsymbol{\gamma}_\theta),$$

to assign the value of $L_2^{(t+1)}$, we need to compute

$$\Pr(L_2|\pi^{(t+1)}, L_2^{(t+1)}, L_3^{(t)}, \dots, L_N^{(t)}, \mathbb{C}, \boldsymbol{\theta}_0^{(t)}, \boldsymbol{\theta}_1^{(t)}; \gamma_{\pi_1}, \gamma_{\pi_0}, \boldsymbol{\gamma}_\theta),$$

and so forth.¹⁷ Intuitively, we're going to pick a bag of words, hold it out from the information we know, then sample for that bag's document label given all the other information in the corpus. We do this repeatedly for each document in turn. Sampling every document once is one iteration.

As mentioned above, we can simplify the sampler by making the continuous-valued parameters π , $\boldsymbol{\theta}_0$, and $\boldsymbol{\theta}_1$ disappear from the calculation. The key is the observation that we can calculate the conditional probability of generating a document of a particular class, given the other documents in the class, by integrating over *all possible word distributions*. Intuitively, this is precisely the same principle as computing the marginal probability for a discrete distribution. If we have an expression for $\Pr(a, b)$, we can compute $\Pr(a)$ by summing over all possible values of b , i.e. $\Pr(a) = \sum_b \Pr(a, b)$. Summing over b is a way of getting rid of it in our expression for $\Pr(a)$. With continuous variables we integrate over all possible values of the variable rather than summing.

¹⁷There's no superscript on the bags of words \mathbf{W}_n because they're fully observed and don't change from iteration to iteration.

2.5.1 Integrating out π

We'll start by integrating out π , and we'll see that the principles we apply here will also come into play again when we integrate out θ , with appropriate differences for θ being a multinomial. Our discussion in this section comes largely from Goldwater's [6] Ph.D. thesis.

Our goal is to calculate the probability of generating a particular label (and document as a consequence) given all of the other labels that have been generated.

$$\Pr(L_j = 1 | \mathbf{L}^{(-j)}; \gamma_{\pi_1}, \gamma_{\pi_0}) = \int \Pr(L_j = 1 | \pi) \Pr(\pi | \mathbf{L}^{(-j)}; \gamma_{\pi_1}, \gamma_{\pi_0}) d\pi \quad (35)$$

$$= \int \pi \frac{\Gamma(N + \gamma_{\pi_1} + \gamma_{\pi_0} - 1)}{\Gamma(C_1 + \gamma_{\pi_1})\Gamma(C_0 + \gamma_{\pi_0})} \pi^{C_1 + \gamma_{\pi_1} - 1} (1 - \pi)^{C_0 + \gamma_{\pi_0} - 1} d\pi \quad (36)$$

$$= \frac{\Gamma(N + \gamma_{\pi_1} + \gamma_{\pi_0} - 1)}{\Gamma(C_1 + \gamma_{\pi_1})\Gamma(C_0 + \gamma_{\pi_0})} \int \pi^{C_1 + \gamma_{\pi_1}} (1 - \pi)^{C_0 + \gamma_{\pi_0} - 1} d\pi \quad (37)$$

$$= \frac{\Gamma(N + \gamma_{\pi_1} + \gamma_{\pi_0} - 1)}{\Gamma(C_1 + \gamma_{\pi_1})\Gamma(C_0 + \gamma_{\pi_0})} \frac{\Gamma(C_1 + \gamma_{\pi_1} + 1)\Gamma(C_0 + \gamma_{\pi_0})}{\Gamma(N - 1 + \gamma_{\pi_1} + \gamma_{\pi_0} + 1)} \quad (38)$$

$$= \frac{C_1 + \gamma_{\pi_1}}{N + \gamma_{\pi_1} + \gamma_{\pi_0} - 1} \quad (39)$$

Note here that we're explicitly leaving L_j out of the calculation since we're looking at all other labels as evidence. This means that either C_0 or C_1 will be one less, and we take this into account before calculating (39), hence the $C_0 + C_1 = N - 1$ simplification. We get to line (36) by substituting the definition of π and by equation (17) above. Since the normalization constant of the Dirichlet distribution does not contain π , it is constant with respect to the integral and we factor it out, then simplify the single π . Notice that the integral in (37) is now of a particularly convenient form—it's a Beta distribution! Since the Beta distribution is a probability distribution, it needs to sum to one, the value of the integral is just its normalization constant (see equations 17 and 18). Substituting the normalization constant in for the integral gets us (38). Finally, we apply the identity $\Gamma(x + 1) = x\Gamma(x)$ to cancel all of the Γ functions, leaving us with a fairly intuitive result: the probability of generating another document with label 1 is the ratio of the number of documents *currently* labeled 1 plus the pseudocounts representing our prior, to the total number of documents and pseudocounts.

2.5.2 Integrating out θ

Our goal here will be to obtain a crucial probability: *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 start first by calculating the probability of generating a specific word given the other words in the class, subtracting out the information about \mathbf{W}_j , so 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,

$$\Pr(w_k = y | \mathbb{C}_x^{(-j)}; \gamma_\theta) = \int_{\Delta} \Pr(w_k = y | \boldsymbol{\theta}) \Pr(\boldsymbol{\theta} | \mathbb{C}_x^{(-j)}; \gamma_\theta) d\boldsymbol{\theta} \quad (40)$$

$$= \int_{\Delta} \theta_{x,y} \frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})} \prod_{i=1}^V \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i} - 1} d\boldsymbol{\theta} \quad (41)$$

$$= \frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})} \int_{\Delta} \theta_{x,y} \prod_{i=1}^V \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i} - 1} d\boldsymbol{\theta} \quad (42)$$

$$= \frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})} \int_{\Delta} \theta_{x,y}^{\mathcal{N}_{\mathbb{C}_x}(y) + \gamma_{\theta_y}} \prod_{i=1 \wedge i \neq y}^V \theta_{x,i}^{\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i} - 1} d\boldsymbol{\theta} \quad (43)$$

$$= \frac{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})}{\prod_{i=1}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})} \frac{\Gamma(\mathcal{N}_{\mathbb{C}_x}(y) + \gamma_{\theta_y} + 1) \prod_{i=1 \wedge i \neq y}^V \Gamma(\mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i})}{\Gamma(\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i} + 1)} \quad (44)$$

$$= \frac{\mathcal{N}_{\mathbb{C}_x}(y) + \gamma_{\theta_y}}{\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x}(i) + \gamma_{\theta_i}} \quad (45)$$

Crucially, here we define $\mathbb{C}_x^{(-j)}$ as the set of all documents in class x *except* for \mathbf{W}_j . Similar to the size of each class from the calculation above, note that since document j is in exactly one of \mathbb{C}_0 or \mathbb{C}_1 , for one class it will be the case that $\mathbb{C}_x^{(-j)} = \mathbb{C}_x$ and for the other $\mathbb{C}_x^{(-j)} = \mathbb{C}_x - \{\mathbf{W}_j\}$. The process we use to integrate out $\boldsymbol{\theta}$ is exactly the same as for integrating out π , though now we're doing it for a multinomial. The set Δ is the probability simplex of $\boldsymbol{\theta}$, namely the set of all $\boldsymbol{\theta}$ such that $\sum_i \theta_i = 1$. We get to (41) by substitution from the formulas we derived in Section 2.4.2, then (42) by factoring out the normalization constant for the Dirichlet distribution, since it is constant with respect to $\boldsymbol{\theta}$. Note that the integrand of (43) is actually another Dirichlet distribution, so its integral is its normalization constant (same reasoning as before). We substitute this in to obtain (44). Using the property of Γ that $\Gamma(x+1) = x\Gamma(x)$ for all x , we can again cancel all of the Γ terms.

Since we assume that the words within a particular document are drawn from the same distribution, we calculate the probability of \mathbf{W}_j by multiplying (45) over all words in the vocabulary.

$$\Pr(\mathbf{W}_j | \mathbb{C}_x^{(-j)}; \gamma_\theta) = \prod_{i=1}^V \frac{\mathcal{N}_{\mathbb{C}_x^{(-j)}}(i) + \gamma_{\theta_i}}{\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x^{(-j)}}(i) + \gamma_{\theta_i}} \quad (46)$$

Voila! No more continuous variables π , $\boldsymbol{\theta}_0$, or $\boldsymbol{\theta}_1$. So, the full expression for the joint distribution is:

$$\Pr(\mathbf{L}, \mathbb{C}; \gamma_{\pi_1}, \gamma_{\pi_0}, \gamma_\theta) = \frac{C_1 + \gamma_{\pi_1}}{N + \gamma_{\pi_1} + \gamma_{\pi_0} - 1} \left(\frac{C_0 + \gamma_{\pi_1}}{N + \gamma_{\pi_1} + \gamma_{\pi_0} - 1} \right) \prod_{j=1}^N \prod_{i=1}^V \prod_{x=0}^1 \frac{\mathcal{N}_{\mathbb{C}_x^{(-j)}}(i) + \gamma_{\theta_i}}{\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_x^{(-j)}}(i) + \gamma_{\theta_i}} \quad (47)$$

We'll actually use *pieces* of this joint distribution in sampling, as you'll see in a moment.

2.5.3 Sampling the conditional distribution

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 $L_j^{(t+1)}$. By definition of conditional probability,

$$\Pr(L_j | \mathbf{L}, \mathbb{C}; \gamma_{\pi_1}, \gamma_{\pi_0}, \gamma_\theta) = \frac{\Pr(L_j, \mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}; \gamma_{\pi_1}, \gamma_{\pi_0}, \gamma_\theta)}{\Pr(\mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}; \gamma_{\pi_1}, \gamma_{\pi_0}, \gamma_\theta)} \quad (48)$$

$$= \frac{\Pr(\mathbf{L}, \mathbb{C}; \gamma_{\pi_1}, \gamma_{\pi_0}, \gamma_\theta)}{\Pr(\mathbf{L}^{(-j)}, \mathbb{C}^{(-j)}; \gamma_{\pi_1}, \gamma_{\pi_0}, \gamma_\theta)} \quad (49)$$

where $\mathbf{L}^{(-j)}$ are all the document labels except L_j , and $\mathbb{C}^{(-j)}$ is the set of all documents except \mathbf{W}_j . The probability that we’re interested in is (47) divided by what you get when you take (47) and remove all information gained from \mathbf{W}_j . (If you find that confusing, refer back to equation 15.)

Luckily for us, most of the terms will cancel! Remember that each and every term in (47) is nothing more than either a count or a prior, so all we need to do is remove word counts from \mathbf{W}_j from the totals, then look at what changes. If we’re considering \mathbf{W}_j as class 0, then none of class 1’s counts change, so we don’t need to consider them (they’ll cancel). Similarly with class 1.¹⁸ Additionally, none of the other documents change while we’re sampling for L_j , so the product over all documents cancels too, *with the exception of* \mathbf{W}_j . This leaves us with the two simpler equations that our sampler will actually compute.

$$\Pr(L_j = 0 | \mathbf{L}, \mathbb{C}; \gamma_{\pi_1}, \gamma_{\pi_0}, \gamma_{\theta}) = \frac{C_0 + \gamma_{\pi_1}}{N + \gamma_{\pi_1} + \gamma_{\pi_0} - 1} \prod_{i=1}^V \frac{\mathcal{N}_{\mathbb{C}_0^{(-j)}}(i) + \gamma_{\theta i}}{\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_0^{(-j)}}(i) + \gamma_{\theta i}} \quad (50)$$

$$\Pr(L_j = 1 | \mathbf{L}, \mathbb{C}; \gamma_{\pi_1}, \gamma_{\pi_0}, \gamma_{\theta}) = \frac{C_1 + \gamma_{\pi_1}}{N + \gamma_{\pi_1} + \gamma_{\pi_0} - 1} \prod_{i=1}^V \frac{\mathcal{N}_{\mathbb{C}_1^{(-j)}}(i) + \gamma_{\theta i}}{\sum_{i=1}^V \mathcal{N}_{\mathbb{C}_1^{(-j)}}(i) + \gamma_{\theta i}} \quad (51)$$

Notice that these equations are *exactly the same*; they differ only by which class we’re examining. When we sample, we’re effectively asking two questions, and then normalizing. The two questions are:

1. “How likely is it that document n is labeled with a 0 using the corpus as evidence?”
2. “How likely is it that document n is labeled with a 1 using the corpus as evidence?”

So, finally, here’s the procedure to sample from the conditional distribution:

- Let value0 = expression (50)
- Let value1 = expression (51)
- Let $p = \frac{\text{value1}}{\text{value0} + \text{value1}}$
- Select the new value for L_j as the result of a Bournoulli trial (coin flip) with success parameter p .

2.5.4 Taking advantage of documents with labels

Briefly: don’t sample L_j for those documents, always keep 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 (50) and (51) and will never be subtracted out.

2.5.5 Putting it all together

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

Sampling iterations.

For $t = 1 \dots T$,

For $j = 1 \dots N$, i.e. for every document,

If document j is not a labeled training document

Remove document j ’s word counts from whatever class it’s a member of

Subtract 1 from the count of documents with label L_j

Assign a new label $L_j^{(t+1)}$ to document j as described at the end of Section 2.5.3.

¹⁸See our earlier footnote about why the subscripts on parameters like θ get dropped.

V	number of words in the vocabulary.
N	number of documents in the corpus.
M_j	number of sentences in document j .
M	number of sentences in the entire corpus.
γ_π	two element column vector containing the hyperparameters for π .
γ_τ	two element column vector containing the hyperparameters for τ_0 and τ_1 .
γ_θ	length V column vector for the word distribution hyperparameters.
W	$N \times V$ matrix where each element W_{ij} is the count of word j in document i .
GP	$N \times 2$ matrix where each element GP_{jp} is the frequency of sentences with perspective label p in document j .
L	vector of length N where each element L_i is the document sentiment label for document i .
S_j	$V \times M_j$ matrix where each S_{jki} is the count of word i in sentence k of document j .

Figure 5: Notation for the discussion of the Latent Sentence Perspective Model (LSPM).

Add 1 to the count of documents with label $L_j^{(t+1)}$
 Add document j 's word counts to the counts for class $L_j^{(t+1)}$

Notice that as soon as a new label for 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 4.

3 Deriving a Gibbs Sampler for a More Complex Model

The Naive Bayes model is surprisingly useful. However, even more interesting models become possible when you introduce additional latent variables into the generative story. A familiar example: hidden Markov models, which explain a sequence of observed symbols in terms of an underlying path through a set of hidden states.

In this section, we'll be talking about an extension to Naive Bayes models for purposes of modeling "perspective", the Latent Sentence Perspective Model (LSPM) introduced by Lin et al. [12]. Lin et al. demonstrate that this model is useful in experiments involving the Bitter Lemons corpus, a collection in which documents about the middle east are written from either an Israeli or Palestinian perspective. (Naturally one could use this model for labeling document sentiment also, or any other binary class distinction.) In addition to restructuring their discussion and breaking things out in a lot more detail, we'll be making extensive modifications to their notation in the hope of increasing clarity for the reader.

For reference, we provide a legend of the notation we'll use in this section as Figure 5.

3.1 LSPM Generative Story

In this model, the generative process involves binary perspective labels for documents (e.g., is this document about middle east politics written from an Israeli or Palestinian perspective?), binary presence-of-perspective labels for sentences (does this sentence carry perspective, or is it neutral with respect to the two classes, e.g. a piece of background information?), and of course the words in the sentences. The parameters are as follows:

- As in the Naive Bayes model described in Section 2, probability π is used to generate sentiment labels for documents. Specifically, π is the probability of picking sentiment label 1 for a document, so $1 - \pi$ is the probability of picking sentiment label 0. The same π is used for every document. We can see this described graphically in Figure 6 since π lies outside of any plate.

- τ is the probability that the sentence being generated bears perspective, as opposed to being non-perspective bearing. We depart from Lin et al’s LSPM here by defining a single τ for each document, as opposed to a τ for each class.¹⁹
- Four vectors θ_{ds} for $d, s \in \{0, 1\} \times \{0, 1\}$. Each vector can be interpreted as a multinomial distribution over the vocabulary: i.e. θ_{ds} is the probability distribution for generating words in the k^{th} sentence of the j^{th} document, when that document’s label is $L_j = d$ and that sentence’s label is $P_{jk} = s$. The i^{th} dimension of θ_{ds} is $\Pr(w_i | L_j = d, P_{jk} = s)$, i.e. the word probability for the i^{th} word on the vocabulary list, when generating such sentences.

For example, θ_{01} is the vector of word probabilities used in generating sentences where the strength-of-perspective is 1 within documents having sentiment label 0. In Lin et al., sentiment label 1 corresponds to the Israeli perspective, so one might expect this distribution to give high probability to Israeli-perspective words like *security* and *defend*.²⁰

Given those parameters, let’s look at the generative process for documents according to this model. The set of documents is generated as follows:

$$\pi \sim \text{Bernoulli}(1, \gamma_\pi) \tag{52}$$

$$\tau_j \sim \text{Bernoulli}(1, \gamma_\tau) \tag{53}$$

$$\theta_{L_j, P_{jk}} \sim \text{Multinomial}(V, \gamma_\theta) \tag{54}$$

$$L_j \sim \text{Bernoulli}(1, \pi) \tag{55}$$

$$P_{jk} \sim \text{Bernoulli}(1, \tau) \tag{56}$$

$$S_{jk} \sim \text{Multinomial}(R_{jk}, \theta) \tag{57}$$

Here $X \sim \text{Distrib}(\phi)$ indicates as usual that values for random variable X are chosen by sampling from distribution Distrib with parameters ϕ .²¹ Although this is a pretty standard way to describe a Bayesian model, let’s break out the details more completely,

For each document j , with j going from 1 to N :

- Pick a sentiment label $L_j = l_j$ (with $l_j = 0$ or 1) by flipping a biased coin with $\Pr(L_j = 1) = \pi$ (equation 55).
- For each sentence k of document j (from 1 to number of sentences document j , M_j):
 - Pick a perspective bearing label P_{jk} 0 or 1 by flipping a biased coin with $\Pr(P_{jk} = 1) = \tau_j$ (equation 56). 1 indicates that the sentence is perspective bearing.
 - Generate the bag S_{jk} of words in the sentence: Independently for each of the R_{jk} word positions w_i , pick a word from the vocabulary with probability $\Pr(w_i | L_j, P_{jk}) = \theta_{L_j, P_{jk}}$ (equation 57).

Observe that the description of the model in (55)-(57), from Lin et al., is so compact as to be confusing to the uninitiated — in fact, the model really cannot be understood properly without simultaneously looking at the graphical representation in Figure 2 of Lin et al. (2006). Without paying attention to that “plate diagram”, it looks to the untrained eye in (56) as though the label for every sentence is governed by the same binomial distribution, parameterized by a single parameter τ . Not so: as noted in the more computationally oriented description of the generative process above, there are N such distributions, one for each document. Which distribution you use for sentence S_{jk} depends on the value of L_j , i.e. the distribution over sentence-level

¹⁹Neither way is particularly more complicated than the other, but we find it less intuitive to link the proportions of perspective bearing sentences in different documents.

²⁰Lin et al. force $\theta_{1,0} = \theta_{0,0}$, arguing that regardless of whether you’re in Israeli or a Palestinian document, sentences not expressing a strong perspective can be expected to have the same distribution over the vocabulary. Presumably this distribution gives higher weight to words that would be used on both sides to express neutral background information.

²¹This part was actually written before the previous section on Naive Bayes, so there are places where we’re re-introducing something basic or being a little repetitive. We’ll fix this in the next revision.

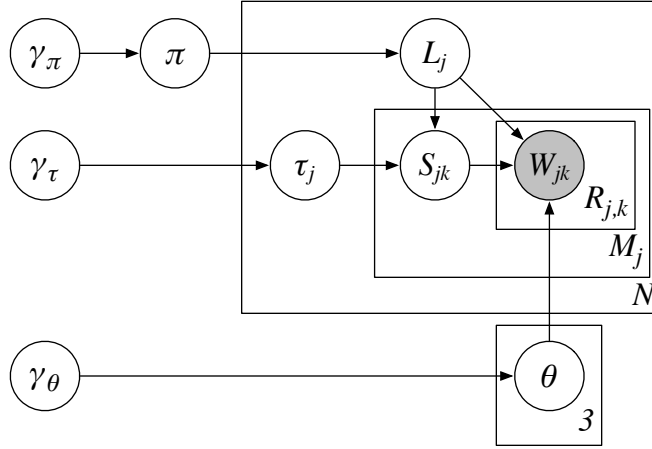


Figure 6: LSPM plate diagram

strength-of-perspective labels depends on the document-level sentiment label for the document the sentence is in. This fact is not reflected anywhere in their formal mathematical description of the model, nor even in the text describing it; it only shows up in the “plate diagram”, which shows that the frequency distribution S_{jk} depends on L_j .

3.2 Expressing the joint distribution

According to our model, the joint distribution is $p(W, L, P, \pi, \tau, \theta; \gamma_\pi, \gamma_\tau, \gamma_\theta)$. It can be expressed in a way that is very similar to the joint distribution in Section 2.4.1. We leave this as an exercise for the reader, though we plan to provide the full form in an upcoming revision of this document.

3.3 Deriving the LSPM Gibbs Sampler Equations

Similarly, we encourage the reader to work out in detail how to derive the sampling steps for this model, the outcome of which is equations (58) and (59). Again, we plan to provide the details in an upcoming revision of this document.²² Crucially, the process will look pretty much identical to what we did with the Gibbs sampler for Naive Bayes.

The sampling steps are:

$$\Pr(S_{jk} = p | \nu^{(-k)}) = \frac{\text{GP}_{jp}^{(-k)} + \gamma_{\tau,p}}{M_j + \gamma_{\tau,0} + \gamma_{\tau,1} - 1} \prod_{i=1}^V \frac{(1-p)W_{ji}^{(-k)} + (p-1)X_i^{(-k)} + pX_i^{(-k)} + \gamma_{\theta,i}}{\sum_{y=1}^V [(W_j - S_k)_y + \gamma_{\theta,y}]}, \quad (58)$$

where we define $\nu^{(-k)}$ to be the set of all variables with the exception of the counts in sentence k of document j , and

$$\Pr(L_j = x | \nu^{(-j)}) = \frac{C_x + \gamma_{\pi,x}}{N + \gamma_{\pi,0} + \gamma_{\pi,1} - 1} \left(\frac{(X_1^{(-j)} + \gamma_{\tau,1})(X_0^{(-j)} + \gamma_{\tau,0})}{M^{(-j)} + \gamma_{\tau,1} + \gamma_{\tau,0}} \right) \prod_{i=1}^V \frac{Y_i^{(-j)} + \gamma_{\theta,i}}{\sum_{y=1}^V Y_y^{(-j)} + \gamma_{\theta,y}} \quad (59)$$

We define $\nu^{(-j)}$ to be the set of all variables with the exception of the counts in the entirety of document j . C_x indicates the total number of documents in the corpus with label x (i.e. $C_1 = \sum_{d=1}^N L_d^{(-j)}$) and

²²Better yet, for this and the joint distribution, we offer a footnote and a beer to the first person who does it correctly themselves and sends it to us in LaTeX (using our notation) so we can just drop it into this document.

$C_0 = N - \sum_{d=1}^N L_d^{(-j)}$. X_0 indicates the total number of perspective bearing sentences in documents labeled 0 (similarly for X_1), and Y_i indicates the frequency of word i in documents with label x . We calculate (59) with $x = 0$ and $x = 1$, then normalize to obtain the probability of assigning label 1 to document j . Notice the similarity between (59) and equations (50) and (51).

4 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 $t = 1, 2, \dots, T$. In theory, the approximated value for any variable Z_i can simply be obtained by calculating:

$$\frac{1}{T} \sum_{t=1}^T z_i^{(t)} \quad (60)$$

However, expression (60) is not always used directly. There are several additional details to note that are a part of typical sampling practice.²³

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 $\langle z_1^{(t)}, z_2^{(t)}, \dots, z_r^{(t)} \rangle$ are all coming from the stationary distribution of the Markov chain, which is an assumption of the approximation in (60). In order to avoid the estimates being contaminated by the values at iterations before that point, we generally discard the values at iterations $t < B$, which are referred to as the “burn-in” iterations, so that the average in (60) is taken only over iterations $B + 1$ through T .²⁴

Autocorrelation and lag. The approximation in (60) 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.²⁵ 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*.²⁶ In this context, Jordan Boyd-Graber (personal communication) also recommends looking at Neal’s [17] discussion of likelihood as a metric of convergence.

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”.²⁷ Lin et al. (2006) run three chains with $T = 5000$ and $B = \frac{T}{2}$. They do not say whether or not they used a lag in order to reduce the effects of autocorrelation, nor do they say how they combined samples from the three chains.

²³Jason Eisner (personal communication) argues, with 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 ([15], end of section 29.9, page 381) and Koller and Friedman ([11], end of section 12.3.5.2, page 522).

²⁴As 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.

²⁵Lohninger [13] observes that “most inferential tests and modeling techniques fail if data is autocorrelated”.

²⁶Again, 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.

²⁷Again, 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.

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. [21] and Escobar and West [3].

5 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 [19] doctoral dissertation has a very nice discussion of parameter estimation in Chapter 4, including a detailed exposition of an EM algorithm for Naive Bayes and his own derivation of a Naive 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 [18]; 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 [8] “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 stand-alone 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” [10], after which you’ll be equipped to look at Johnson, Griffiths, and Goldwater [9].²⁸ 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 (15) 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 2.5.3–2.5.5) 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 pick probabilistically based on the result, and finally add counts back in according to the probabilistic choice you just made.

²⁸As 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. [7]. Johnson et al. don’t appear to have been aware of this in their seminal paper introducing Gibbs sampling for PCFGs to the NLP community, and a search on scholar.google.com turned up no citations in the NLP literature.

If you find errors in this document, please send them to us at resnik@umiacs.umd.edu. We would also be very happy to hear about other resources that would be helpful for people new to this topic.

Acknowledgments

The creation of this document has been supported in part by NSF award IIS-0838801, IARPA contract W911NF09C0129, and the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors. The authors are grateful to Jordan Boyd-Graber, Chris Dyer, Jason Eisner, Kevin Knight, and Prithvi Sen for their helpful comments and discussion. Extra thanks to Jordan Boyd-Graber for assistance with plate diagrams.

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] S. J. Goldwater. *Nonparametric Bayesian Models of Lexical Acquisition*. PhD thesis, Brown University, 2007.
- [7] 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.
- [8] 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>.
- [9] 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.
- [10] 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>.
- [11] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
- [12] W.-H. Lin, T. Wilson, J. Wiebe, and A. Hauptmann. Which side are you on? identifying perspectives at the document and sentence levels. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, 2006.
- [13] H. Lohninger. *Teach/Me Data Analysis*. Springer-Verlag, 1999. http://www.vias.org/tmdatanaleng/cc_corr_auto_1.html.

- [14] A. Lopez. Statistical machine translation. *ACM Computing Surveys*, 40(3):1–49, August 2008.
- [15] D. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [16] C. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
- [17] 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>.
- [18] T. Pedersen. Knowledge lean word sense disambiguation. In *AAAI/IAAI*, page 814, 1997.
- [19] T. Pedersen. *Learning Probabilistic Models of Word Sense Disambiguation*. PhD thesis, Southern Methodist University, 1998. <http://arxiv.org/abs/0707.3972>.
- [20] S. Theodoridis and K. Koutroumbas. *Pattern Recognition, 4th Ed.* Academic Press, 2009.
- [21] 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.