Code-Breaking, Bayes' Rule, and Metropolis-Hastings

Bryan S. Graham, UC - Berkeley & NBER

October 26, 2025

Imagine that, upon completing your Ph.D., you find employment as a secret agent. While on a top secret mission you intercept what appears to be a coded message. You are in a wilderness area, indeed a deep river canyon with heavy tree cover, with no ability to communicate with your handlers. All that is available to you is your acquired human capital and a fully-charged smart phone with a Python IDE installed and a complete text file of *Dune* downloaded on it. This lecture note will provide the information you need to use these tools to decrypt the message and complete your mission. Lets go!

We will assume that the intercepted message was encrypted with a *keyed substitution cipher*. Specifically the original *plaintext* was transformed into a *ciphertext* by replacing each letter in the plaintext alphabet with a unique symbol in a corresponding ciphertext alphabet. To decrypt the message you need to find the *cipher key*, ϕ. The cipher defines a one-to-one (i.e., bijective) mapping from the *ciphertext* alphabet, $\mathbb{B}$, to the *plaintext* alphabet $\mathbb{A}$:

$$\phi_0 : \mathbb{B} \mapsto \mathbb{A}. \tag{1}$$

Note that the sets $\mathbb{A}$ and $\mathbb{B}$ may coincide, but this is not required (i.e., one might be roman letters, the other pictographs). The cardinality of $\mathbb{A}$ and $\mathbb{B}$ *do* coincide. There are $L = J!$ possible keys, where $J = |\mathbb{A}|$ is the number of letters in the plaintext alphabet ($|\mathbb{A}|$ is notation for the cardinality of set $\mathbb{A}$).

Let $\mathbf{y}$ denote the the ciphertext in hand. We'll use the notation $\tilde{\mathbf{y}} = \phi(\mathbf{y})$ to denote corresponding unencrypted plaintext. A simple example helps clarify the notation and basic set-up. Consider a plaintext alphabet of $\mathbb{A} = \{a, b, c\}$ and a corresponding ciphertext alphabet of $\mathbb{B} = \{\alpha, \beta, \gamma\}$. Let the cipher key be

$$\phi(\alpha) = a, \phi(\beta) = b, \phi(\gamma) = c.$$

An alternative, more compact, notation is $\phi = \{(\alpha, a), (\beta, b), (\gamma, c)\}$. Let $\mathbf{y} = \alpha\alpha\gamma\beta\alpha\gamma$ be

the ciphertext in hand, then $\phi(\mathbf{y}) = \tilde{\mathbf{y}} = aacbac$. Similarly $\phi^{-1}(\tilde{\mathbf{y}}) = \alpha\alpha\gamma\beta\alpha\gamma$.

Our target parameter is ϕ. The parameter space Φ is the set of all possible decryption ciphers. This set is discrete and finite, but with $L \overset{def}{=} |\Phi| = J!$ elements, it is massive:

$$\phi_0 \in \Phi = \{\phi_1, \ldots, \phi_L\}.$$

A probability model for plaintexts

When encountering an odd piece of data it is helpful to consider an "as if" sampling process which describes how the data might have come into your hands. A well-defined random sampling process can inform the construction of a probability model – a *likelihood* – for the data. In order to construct a probability model for the *ciphertext* in hand, $\mathbf{y}$, we imagine that is corresponds to an encrypted version of an English *plaintext* sampled at random from a population of English plaintexts. Let $\tilde{\mathbf{Y}}$ denote a random draw from our hypothetical population of plaintexts, then $\mathbf{Y} = \phi^{-1}\left(\tilde{\mathbf{Y}}\right)$ equals the corresponding ciphertext random draw; this is what is observed by the econometrician.

We observe the event $\mathbf{Y} = \mathbf{y}$, from this event we hope to learn something about $\phi \in \Phi$, the unknown substitution cipher key that correctly decodes$\mathbf{y}$. To recap: the thought experiment is that the "enemy encryptor" takes a random draw from the population of English plaintexts, denoted by $\tilde{\mathbf{Y}}$, and reports to the econometrician the ciphertext $\mathbf{Y} = \phi^{-1}\left(\tilde{\mathbf{Y}}\right)$. This is a stretch given the hypothetical predicament introduced above, but it is a useful fiction to help us get going. Under this fiction, the (*ex ante*) likelihood of the event $\mathbf{Y} = \mathbf{y}$ – *when* ϕ *is the encryption cipher* – coincides with the (*ex ante*) likelihood that, $\tilde{\mathbf{Y}}$, a randomly sampled plaintext equals $\phi(\mathbf{y})$. This follows because the cipher is bijective: fixing the cipher at a hypothesized null value, here ϕ, the mapping from a plaintext to a ciphertext is one-to-one. Hence – conditional on the cipher, ϕ, the probability of the event

$$\begin{aligned}\Pr\left(\tilde{\mathbf{Y}} = \tilde{\mathbf{y}}\right) &= \Pr\left(\tilde{\mathbf{Y}} = \phi(\mathbf{y})\right) \\ &= \Pr\left(\phi^{-1}\left(\tilde{\mathbf{Y}}\right) = \phi^{-1}(\phi(\mathbf{y}))\right) \\ &= \Pr(\mathbf{Y} = \mathbf{y}|\,\phi). \end{aligned} \tag{2}$$

The second equality follows from ϕ being bijective. The likelihood $\Pr\left(\tilde{\mathbf{Y}} = \tilde{\mathbf{y}}\right) = \Pr(\mathbf{Y} = \mathbf{y}|\,\phi)$ raises nontrivial conceptual and practical issues. For now assume we have such a likelihood and the means to evaluate it for any $\phi \in \Phi$. The maximum likelihood estimate (MLE), $\hat{\phi}$, is simply the element of Φ which maximizes the ex ante probability of observing the event

$\mathbf{Y} = \mathbf{y}$ (i.e., the observed ciphertext in hand). Unfortunately direct computation of the MLE requires $L = J!$ likelihood evaluations. This not feasible. Even if we were able to compute the MLE, it is not clear how we could measure, or even express, our uncertainty about how close $\hat{\phi}$ is to ϕ_0 (here ϕ_0 denotes the true cipher used to encrypt the message). All that is available is a *single* randomly sampled ciphertext.

Instead we'll proceed using Bayesian methods. We have no reason, *a priori*, to believe one cipher key is more likely to have been used than any other. This motivates a discrete uniform *prior* of $\Pr(\phi_0 = \phi) \overset{def}{\equiv} \pi(\phi) = \frac{1}{L}$ for all $\phi \in \Phi$. By *Bayes' Rule* the posterior probability that the true cipher is equal to ϕ, $\Pr(\phi_0 = \phi | \mathbf{Y} = \mathbf{y}) \overset{def}{\equiv} \pi(\phi | \mathbf{y})$, equals

$$\begin{aligned} \pi(\phi | \mathbf{y}) &= \frac{\Pr(\mathbf{y} | \phi)\, \pi(\phi)}{\Pr(\mathbf{y})} \\ &= \frac{\Pr(\mathbf{y} | \phi)\, \frac{1}{L}}{\frac{1}{L}\sum_{l=1}^{L} \Pr(\mathbf{y} | \phi_l)} \\ &= \frac{\Pr(\mathbf{y} | \phi)}{\sum_{l=1}^{L} \Pr(\mathbf{y} | \phi_l)}, \end{aligned} \tag{3}$$

where we use the notation $\Pr(z | x) = \Pr(Z = z | X = x)$.

Unfortunately (3) is as difficult to compute as the MLE. In particular, the denominator involves a summation of the likelihood evaluated over all elements in Φ. Denominators of this type are called "intractable normalizing constants" in statistics. If we could efficiently compute this denominator, then we could also compute the MLE, $\hat{\phi}$. Fortunately, as we will show below, it is possible to construct a random draw from the posterior. This will suffice for our purposes.

A composite likelihood

Computing the ex ante probability that a randomly sampled plaintext, $\tilde{\mathbf{Y}}$, takes a particularly configuration is hard. The English language is complicated: a plaintext is not simply a concatenation of independent draws from the alphabet (with letters drawn uniformly at random). Instead of trying to compute probabilities for an *entire* plaintext taking a particularly form, consider instead doing this for a randomly sampled pair of consecutive letters – a *bigram* – in a plaintext. Let $\tilde{Y}$ be the configuration of a randomly sampled bigram from a representative (i.e., randomly sampled) English plaintext. A *bigram* is an ordered sequence of two letters:

$$\tilde{Y} \in \mathbb{A} \times \mathbb{A}.$$

Note that consecutive bigrams with the same word overlap. For example, the word "DUNE" consists of the bigrams "DU", "UN" and "NE".

There are a total of J^2 possible bigrams. Let $\pi_{aa'}$ be the probability that a randomly sampled English bigram equals aa'; $\tilde{Y}$ is a multinomial random variable with likelihood

$$\Pr\left(\tilde{Y}=\tilde{y}\right)=\prod_{a\in\mathbb{A}}\prod_{a'\in\mathbb{A}}\pi_{aa'}^{\mathbf{1}(\tilde{y}=aa')}. \tag{4}$$

Under the hypothesis that the true substitution cipher is ϕ, the probability of observing that a randomly sampled bigram, Y, in our ciphertext equals y is

$$\Pr\left(Y=y|\,\phi\right)=\prod_{a\in\mathbb{A}}\prod_{a'\in\mathbb{A}}\pi_{aa'}^{\mathbf{1}(\phi(y)=aa')}. \tag{5}$$

The probability $\pi_{aa'}$ is well defined. Enumerate all plaintexts in the universe, sample a bigram at random from these texts, $\pi_{aa'}$ is the ex ante probability that the sampled bigram equals aa'.

Our cipher text consists of $k=1,\ldots,N$ encrypted bigrams. Unfortunately these bigrams are not sampled independently at random from the population of plaintext bigrams. Instead these bigrams appear sequentially within a single text. The word "the" is a common word in English. If we observe the bigram "TH" in a plaintext, we might expect the chance that it is followed by "HE" to be higher than the unconditional probability of sampling "HE".

The *composite likelihood* for the ciphertext $\mathbf{y}=(y_1,y_2,\ldots,y_N)$ is

$$\begin{aligned}\ell_N^{\text{cl}}\left(\phi\right)&=\prod_{k=1}^{N}\Pr\left(Y=y_k|\,\phi\right)\\&=\prod_{k=1}^{N}\left\{\prod_{a\in\mathbb{A}}\prod_{a'\in\mathbb{A}}\pi_{aa'}^{\mathbf{1}(\phi(y_k)=aa')}\right\}.\end{aligned} \tag{6}$$

This is a composite likelihood, not a true likelihood, since its multiplicands are not independent of each other. To reiterate: a ciphertext does not consist of an simple random sample of encrypted bigrams from the population of English bigrams. Our composite likelihood is constructed from the *sequence* of encrypted bigrams found in the ciphertext in hand. Bigrams drawn from a single text will be dependent, particularly if they are nearby one another. Our composite likelihood is written as if these sequentially sampled bigrams are independent of one another.

Although each multiplicand in (6) corresponds to a correct marginal probability for the event that randomly sample bigram Y is such that $Y=y_k$. The expression as a whole is *not* the

correct *joint* probability for the entire sequence of bigrams observed in our ciphertext. We'll ignore this issue in the interest of pushing ahead.

To compute $\pi_{aa'}$ we could look at the entire universe of English plaintexts in our target population and compute the frequency with which the bigram aa' appears in this reference set. In practice it is more practical to just work with a reference plaintext (e.g., a single novel) and compute:

$$\pi_{aa'} = \frac{\#\text{ of times bigram } aa' \text{ appears in reference text}}{\text{total number of bigrams in reference text}}$$

Since we have a copy of Frank Herbert novel *Dune* on our smartphone, we'll use the frequency of the bigram aa' in this text to approximate $\pi_{aa'}$.

Define the "composite posterior":

$$\pi^{\text{cp}}\left(\phi|\,\mathbf{y}\right) \overset{def}{\equiv} \frac{\ell_N^{\text{cl}}\left(\phi\right)\pi\left(\phi\right)}{\sum_{l=1}^{L}\ell_N^{\text{cl}}\left(\phi_l\right)\pi\left(\phi_l\right)} = \frac{\ell_N^{\text{cl}}\left(\phi\right)}{\sum_{l=1}^{L}\ell_N^{\text{cl}}\left(\phi_l\right)}.$$

Formally we can not use $\pi^{\text{cp}}\left(\phi|\,\mathbf{y}\right)$ to make posterior probability statements (since our likelihood is not correct), but in what follows we'll just treat it like a regular posterior probability mass function (pmf) and hope for the best.

Metropolis-Hastings Algorithm

Our goal is to construct an method for taking random draws from $\pi^{\text{cp}}\left(\cdot|\,\mathbf{y}\right)$. We begin by choosing some $\phi^{(0)} \in \Phi$ as an initial cipher guess. Given $\phi^{(0)}, \phi^{(1)}, \ldots, \phi^{(s)}$ we generate the next cipher guess $\phi^{(s+1)}$as follows

1. Construct a proposal cipher $\tilde{\phi}^{(s+1)}$ by randomly transposing the plaintext values the current cipher $\phi^{(s)}$ assigns to any two symbols in the *ciphertext* alphabet $\mathbb{B}$. We sample from $\mathbb{B}$ uniformly at random (with replacement). For example, we would go from$\phi^{(s)} = \{(\alpha, a), (\beta, b), (\gamma, c)\}$ to $\tilde{\phi}^{(s+1)} = \{(\alpha, b), (\beta, a), (\gamma, c)\}$ with probability $\frac{2}{9}$. Because we sample with replacement, there is some chance that the same letter is sampled twice, in which case $\tilde{\phi}^{(s+1)} = \phi^{(s+1)}$. In our simple example this occurs $\frac{1}{3}$ of the time (check this claim).

2. We set $\phi^{(s+1)} = \tilde{\phi}^{(s+1)}$ with probability $\min\left\{1, \frac{\pi\left(\tilde{\phi}^{(s+1)}|\mathbf{y}\right)}{\pi\left(\phi^{(s)}|\mathbf{y}\right)}\right\}$. This means we *accept* any proposal with a higher posterior likelihood, while accepting proposals with lower

posterior likelihood with positive probability (but less than 1). This feature of the algorithm means that we will spend more time exploring cipher keys with higher posterior probability. After this step we return to step 1.

It turns out that, if we let our algorithm continue long enough, $\phi^{(s+1)}$ will coincide with a random draw from our "composite posterior":

$$\phi^{(s+1)} \sim \pi\left(\cdot \middle| \mathbf{y}\right).$$

Concretely this means that if our sequence repeatedly visits some cipher $\phi^* \in \Phi$ much more often than any other, then our posterior beliefs that this is the correct cipher are high (i.e., we believe with high probability that ϕ^* was used to encrypt our message).

Implementation details and comments

Note that although $\pi\left(\phi^{(s)} \middle| \mathbf{y}\right)$ and $\pi\left(\tilde{\phi}^{(s+1)} \middle| \mathbf{y}\right)$ are difficult to evaluate separately, evaluating their ratio is easy since

$$\frac{\pi\left(\tilde{\phi}^{(s+1)} \middle| \mathbf{y}\right)}{\pi\left(\phi^{(s)} \middle| \mathbf{y}\right)} = \frac{\ell_N^{\text{cl}}\left(\tilde{\phi}^{(s+1)}\right)}{\ell_N^{\text{cl}}\left(\phi^{(s)}\right)}.$$

This is a key selling point of the Metropolis-Hasting algorithm.
Let $U^{(s)} \sim \mathcal{U}\left[0,1\right]$. In step 2 we set $\phi^{(s+1)} = \tilde{\phi}^{(s+1)}$ (i.e., we accept the proposed cipher) if

$$\frac{\ell_N^{\text{cl}}\left(\tilde{\phi}^{(s+1)}\right)}{\ell_N^{\text{cl}}\left(\phi^{(s)}\right)} \geq U^{(s)}$$

and set $\phi^{(s+1)} = \phi^{(s)}$ otherwise. In practice it is easier to work with the log difference:

$$\ln \ell_N^{\text{cl}}\left(\tilde{\phi}^{(s+1)}\right) - \ln \ell_N^{\text{cl}}\left(\phi^{(s)}\right)$$

and choose $\phi^{(s+1)}$ according to the rule

$$\phi^{(s+1)} = \begin{cases} \tilde{\phi}^{(s+1)} & \text{if } -\left[\ln \ell_N^{\text{cl}}\left(\tilde{\phi}^{(s+1)}\right) - \ln \ell_N^{\text{cl}}\left(\phi^{(s)}\right)\right] \leq V^{(s)} \\ \phi^{(s)} & \text{otherwise} \end{cases}$$

with $V^{(s)} \sim \text{Exponential}\left(1\right)$. This follows by observing that if $U^{(s)} \sim \mathcal{U}\left[0,1\right]$, then $-\ln U^{(s)} \sim \text{Exponential}\left(1\right)$.

Observe that the composite log-likelihood equals

$$\ln \ell_N^{\text{cl}}\left(\phi^{(s)}\right) = \sum_{k=1}^{n} \sum_{a\in\mathbb{A}} \sum_{a'\in\mathbb{A}} \mathbf{1}\left(\phi^{(s)}\left(y_k\right) = aa'\right) \ln \pi_{aa'}.$$

To avoid numerical underflow issues is often helpful to replace $\ln \pi_{aa'}$ with $\max\left(\ln \pi_{aa'}, \ln \epsilon\right)$ with ϵ a (very) small positive number. For example, the bigram 'QK' effectively does not appear in English, but it may be that $\phi^{(s)}\left(y_k\right) =$ 'QK' for some candidate cipher key. If we set $\pi_{\text{QK}} = 0$ we'll get a "log of zero" error.

Some theory

Our algorithm moves randomly through the parameter space, Φ (the set of all $L = J!$ possible decryption ciphers). This space is discrete and finite. We say that $\phi' \in \Phi$ is a *neighbor* of $\phi \in \Phi$ (with $\phi \neq \phi'$) if it is possible to move from cipher ϕ to cipher ϕ' in a single step of our algorithm. Let $\mathcal{N}\left(\phi\right)$ denote the set of ciphers adjacent to ϕ (i.e., reachable in a single step). Note $\left|\mathcal{N}\left(\phi\right)\right| = \frac{1}{2}J\left(J-1\right)$ for all $\phi \in \Phi$.

At each iteration of the algorithm we make a random transposition of the plaintext alphabet values ϕ assigns to two symbols in the *ciphertext* alphabet $\mathbb{B}$. Since there are J symbols in $\mathbb{B}$ and we sample twice from $\mathbb{B}$ *with replacement*, the probability of considering cipher $\phi' \neq \phi$ (with $\phi' \in \mathcal{N}\left(\phi\right)$) is $\frac{2}{J^2}$. Since we draw any letter to swap with probability $\frac{1}{J}$, we sample any *ordered* pair with probability $\frac{1}{J^2}$; but when proposing to go from ϕ to ϕ' which of the two orders at which the swapped pair is drawn does not matter. The algorithm will also propose to move from ϕ to ϕ (i.e, 'propose' to stay in the same place) with probability $\frac{J}{J^2} = \frac{1}{J}$ (check this calculation).

If, upon considering the ϕ-to-ϕ' transition, we have $\pi\left(\phi'|\,\mathbf{y}\right) \geq \pi\left(\phi|\,\mathbf{y}\right)$, we implement the transition with probability one. Otherwise we implement the transition with probability $\frac{\pi(\phi'|\mathbf{y})}{\pi(\phi|\mathbf{y})} < 1$. We stay at parameter ϕ when (i) we sample the same symbol from $\mathbb{B}$ twice or (ii) we reject a candidate move from ϕ-to-ϕ' .

Let $\mathbf{P} = \left[p_{\phi\phi'}\right]_{\phi,\phi'\in\Phi}$ be the $L \times L$ transition matrix characterizing our Markov chain. The elements of this matrix are given by

$$p_{\phi\phi'} = \begin{cases} \frac{2}{J^2}\min\left\{1, \frac{\pi(\phi'|\mathbf{y})}{\pi(\phi|\mathbf{y})}\right\} & \text{if } \phi \neq \phi' \text{ and } \phi' \in \mathcal{N}\left(\phi\right) \\ 0 & \text{if } \phi \neq \phi' \text{ and } \phi' \notin \mathcal{N}\left(\phi\right) \\ 1 - \sum_{\phi'\neq\phi} p_{\phi\phi'} & \text{if } \phi = \phi' \end{cases}. \tag{7}$$

Note that this matrix has lots of zeros since not all $\phi' \in \Phi$ are neighbors of ϕ. The addition of the self-loop is important. It makes sure the algorithm spends "enough time" at state ϕ.

Because our algorithm can move between any two elements of Φ in a finite number of iterations it is *irreducible* (see Definition 7.3 on p. 164 in Mitzenmacher & Upfal (2005)). A second important property of our Markov Chain is that if it visits ϕ, then it will eventually return to ϕ with probability one. This means that each element of Φ is visited infinitely often if the chain runs forever. Formally our Markov chain is *recurrent* (see Definition 7.4 on p. 164 in Mitzenmacher & Upfal (2005)). We can also show that our chain is *aperiodic*. Informally this means that the chain does not cycle through any regularly, repeating sequences of cipher guesses (see Definitions 7.6 and 7.7 on pp. 164-165 of Mitzenmacher & Upfal (2005)).

Because any finite, irreducible and aperiodic Markov chain is ergodic, meaning it has a unique stationary distribution that converges to over time (i.e., after enough iterations), we can apply the following theorem.

Theorem 1. *(Stationary Distribution) Consider a finite, irreducible, and ergodic Markov chain with L states and transition matrix $\mathbf{P}$. If (i) there exists a vector of non-negative numbers $\underline{\pi} = (\pi_1, \ldots, \pi_L)'$ such that (i) $\sum_{l=1}^{L} \pi_l = 1$ and (ii) $\pi_l p_{lm} = \pi_m p_{ml}$ for any pair $l, m = 1, \ldots, L$, then $\underline{\pi}$ is the stationary distribution of the Markov chain with transition matrix $\mathbf{P}$.*

Theorem 1 is standard and available in many textbooks (e.g., Theorem 7.10 on p. 172 of Mitzenmacher & Upfal (2005)).

We can use Theorem 1 to verify that the stationary distribution of our Markov change coincides with the target posterior distribution of interest. Finiteness, irreducibility and ergodicity have already been discussed. Condition (i) of the Theorem is satisfied by construction (see Equation (7)). Requirement (ii) is typically referred to as the *detailed balanced* condition.

To verify condition (ii) let ϕ_l and ϕ_m be two elements in Φ labelled such that $\pi(\phi_l|\mathbf{y}) \geq \pi(\phi_m|\mathbf{y})$ (without loss of generality) and work with the following isomorphism to the objects in the statement of Theorem 1: $\pi_l = \pi(\phi_l|\mathbf{y})$, $\pi_m = \pi(\phi_m|\mathbf{y})$, $p_{lm} = \frac{2}{J^2} \min\left\{1, \frac{\pi(\phi_m|\mathbf{y})}{\pi(\phi_l|\mathbf{y})}\right\}$ and $p_{ml} = \frac{2}{J^2} \min\left\{1, \frac{\pi(\phi_l|\mathbf{y})}{\pi(\phi_m|\mathbf{y})}\right\}$. If ϕ_l and ϕ_m are such that $\phi_l \notin \mathcal{N}(\phi_m)$ or $\phi_m = \phi_l$, then

condition (ii) trivially holds, otherwise, for $\phi_l \neq \phi_m$:

$$\begin{aligned}
\pi_l p_{lm} =& \pi\left(\phi_l|\,\mathbf{y}\right)\frac{2}{J^2}\min\left\{1,\frac{\pi\left(\phi_m|\,\mathbf{y}\right)}{\pi\left(\phi_l|\,\mathbf{y}\right)}\right\}\\
=& \pi\left(\phi_l|\,\mathbf{y}\right)\frac{2}{J^2}\frac{\pi\left(\phi_m|\,\mathbf{y}\right)}{\pi\left(\phi_l|\,\mathbf{y}\right)}\\
=& \pi\left(\phi_m|\,\mathbf{y}\right)\frac{2}{J^2}\\
=& \pi\left(\phi_m|\,\mathbf{y}\right)\frac{2}{J^2}\min\left\{1,\frac{\pi\left(\phi_l|\,\mathbf{y}\right)}{\pi\left(\phi_m|\,\mathbf{y}\right)}\right\}\\
=& \pi_m p_{ml}
\end{aligned}$$

as required. The second equality follows from $\min\left\{1,\frac{\pi(\phi_m|\mathbf{y})}{\pi(\phi_l|\mathbf{y})}\right\}=\frac{\pi(\phi_m|\mathbf{y})}{\pi(\phi_l|\mathbf{y})}$ and the fourth from $\min\left\{1,\frac{\pi(\phi_l|\mathbf{y})}{\pi(\phi_m|\mathbf{y})}\right\}=1$ (both implications of $\pi\left(\phi_l|\,\mathbf{y}\right)\geq\pi\left(\phi_m|\,\mathbf{y}\right)$). This gives a stationary distribution for our Markov chain of

$$\underline{\pi}\left(\mathbf{y}\right)=\left(\pi\left(\phi_1|\,\mathbf{y}\right),\ldots,\pi\left(\phi_L|\,\mathbf{y}\right)\right)'$$

as claimed.

Further Reading

An accessible and well-written reference on discrete time Markov Chain theory with countable state spaces is given by Mitzenmacher & Upfal (2005). This lecture material was inspired by Diaconis (2009) and the MA thesis by Connor (2003). Lindsey (1988) a good initial reference on composite likelihood, a tool that appears frequently in applied work (e.g., anytime a researcher uses "clustered" standard errors). Chib & Greenberg (1995) and Minh & Minh (2015) provide nice discussions of the Metropolis-Hasting algorithm. The Wikipedia entry provides some interesting historical background.

References

Chib, S. & Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *American Statistician*, 49(4), 327 – 335.

Connor, S. (2003). Simulation and solving substitution codes. Master's thesis, University of Warwick, Department of Statistics.

Diaconis, P. (2009). The markov chain monte carlo revolution. *Bulletin of the American Mathematical Society*, 46(2), 179 – 205.

Lindsey, B. G. (1988). Composite likelihood. *Contemporary Mathematics*, 80, 221 – 239.

Minh, D. D. L. & Minh, D. L. P. (2015). Understanding the hastings algorithm. *Communications in Statistics - Simulation and Computation*, 44(2), 332 – 349.

Mitzenmacher, M. & Upfal, E. (2005). *Probability and Computing*. Cambridge: Cambridge University Press.

