Supplementary Material for Diachrony-aware Induction of Binary Latent Representations from Typological Features

Yugo Murawaki
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
murawaki@i.kyoto-u.ac.jp

S.1 Modeling the Binary Latent Matrix

In Bayesian statistics, the natural modeling choice for binary latent matrices is an Indian buffet process (IBP) (Griffiths and Ghahramani, 2011). In a preliminary study, we also modeled $P(Z \mid A)$ using an IBP. However, we abandoned this approach for several reasons.

The IBP is a stochastic process that can informally be seen as an extension of the Chinese restaurant process (CRP). While a CRP represents each object by a single latent class, an IBP assigns a sequence of binary parameters to each object. The applications of the IBP include binary matrix factorization (Meeds et al., 2007), independent component analysis (Knowles and Ghahramani, 2007), choice behavior (Görür et al., 2006), and optimality theory (Doyle et al., 2014).

In our case, each language is represented as a sequence of binary latent parameters, and $\tilde{\Theta} = ZW$ encodes inter-feature dependencies. A large number of weights are not easy to optimize because a naïve Metropolis-Hastings algorithm suffers from extremely slow convergence (Neal, 2000). This problem can be addressed by the use of Hamiltonian Monte Carlo because it can efficiently sample a large number of continuous variables at once. See Section S.4 for details.

The IBP is appealing for its ability to adjust the number of parameters to data. Parameter k is called an *active* parameter if it has one or more languages with $z_{l,k} = 1$. The crux of the IBP is that although there are an unbound number of parameters, the number of active parameters K^+ is finite. K^+ changes during posterior inference. It is decremented when one parameter becomes inactive. Similarly, it is incremented when $z_{l,k}$ is turned from 0 to 1 for inactive parameter k.

The first reason for the disuse of the IBP is the difficulty of extending the model to incorporate inter-language dependencies. It appears that we have no choice but to replace the IBP with the product of the autologistic models.

Second, the nonparametric model's adaptability did not work in our case. In theory, Gibbs sampling converges to a stationary distribution after a sufficiently large number of iterations. However, we observed that the number of active parameters heavily depended on its initial value K_0 because it was very rare for additional parameters to survive as active parameters. For this reason, the number of parameters for the present model is given a priori and is fixed throughout posterior inference.

A possible solution is the split-merge sampling scheme (Jain and Neal, 2007). This algorithm proposes to split one parameter into two and conversely to merge two into one. These large-scale moves are accompanied by several Gibbs sampling scans to make the proposals more reasonable. Due to a large number of weights, Gibbs sampling scans for our model is computationally expensive, however.

Lastly, the IBP makes the binary matrix Z too sparse. To see this, let us consider the stick-breaking representation of the IBP (Teh et al., 2007). We first obtain a sequence of $\mu_{(k)}$'s by repeatedly applying the following procedure:

$$\nu_{(k)} \sim \text{Beta}(\alpha, 1) \quad \mu_{(k)} = \prod_{l=1}^{k} \nu_{(l)}. \tag{S.1}$$

Note that $\mu_{(k)}$'s are arranged in a strictly decreasing ordering: $\mu_{(1)} > \mu_{(2)} > \cdots > \mu_{(K)}$. We permute $\mu_{(k)}$'s to obtain μ_k's and then draw $z_{l,k}$ from $\text{Bernoulli}(\mu_k)$. Eq. (S.1) indicates that $\mu_{(k)}$ decays fairly fast, resulting in a small fraction of Z having value 1.

A workaround we found is to apply simulated annealing to the sampling of $z_{l,k}$. The annealing let the posterior deviate from the prior, and thanks to slow mixing of Gibbs sampling, Z remained relatively dense throughout posterior inference. A more straightforward solution is to introduce power-law behavior to the IBP (Teh and Görür, 2009), which is analogous to the two-parameter extension to the CRP (Teh, 2006).

S.2 Comparison with P&P Parameters

We would like to make five remarks on differences between the principles and parameters (P&P) framework and the proposed model. Although our binary latent representations are partly inspired by P&P, their differences cannot be ignored. We do not intend to present the proposed method as a computational procedure to induce P&P parameters.

First, while the primary focus of generative linguists is put on morphosyntactic characteristics of languages, the dataset we used in the experiments is not limited to them.

Second, Baker (2002) presented a hierarchical organization of parameters (see Figure 6.4 of Baker (2002)). However, we assume independence between parameters, except for inter-language dependencies. Introducing some hierarchical structure to parameters is an interesting direction to explore, but we leave it for future work.

Third, while P&P hypothesizes deterministic generation, the proposed model *stochastically* generates a language's features from its parameters. This choice appears to be inevitable because obtaining exceptionless relations from real data is virtually impossible.

Fourth, a P&P parameter typically controls a very small set of features. By contrast, if our parameter is turned on, it more or less modifies all the feature generation probabilities. However, we can expect a small number of parameters to dominate the probabilities because weights are drawn from a heavy-tailed distribution. Alternatively, we could explicitly impose sparsity by generating another binary matrix Z_W and replacing W with $Z_W \odot W$, where $\odot$ denotes element-wise multiplication.

Lastly, our parameters are asymmetric in the sense that parameters do not operate at all if they are off, but this is not necessarily the case with P&P. Although asymmetry comes about as a natural result of matrix decomposition involving binary variables, it sacrifices transparency between the two types of parameters. If the two values of a P&P parameter have a marked-unmarked relation, the proposed model would need only one parameter for the marked value. Otherwise, two (or more) parameters would be required to represent a P&P parameter.

S.3 Approximate Sampling from Doubly-intractable Distributions

During inference, we want to sample v_k (and h_k and u_k) from its posterior distribution, $P(v_k \mid -) \propto P(v_k)P(\mathrm{z}_{*,k} \mid v_k, h_k, u_k)$. Unfortunately, we cannot apply the standard Metropolis-Hastings (MH) sampler to this problem because $P(\mathrm{z}_{*,k} \mid v_k, h_k, u_k)$ contains an intractable normalization term. Such a distribution is called a *doubly-intractable* distribution because MCMC itself approximates the intractable distribution (Møller et al., 2006; Murray et al., 2006). This problem remains an active topic in the statistics literature to date. However, if we give up theoretical rigorousness, it is not difficult to draw samples from the posterior, which are only approximately correct but work well in practice.

Specifically, we use the double MH sampler (Liang, 2010). The key idea is to use an auxiliary variable to cancel out the normalization term. This sampler is based on the exchange algorithm of Murray et al. (2006), which samples v_k in the following steps.

1. Propose $v'_k \sim q(v'_k | v_k, \mathrm{z}_{*,k})$.
2. Generate an auxiliary variable $\mathrm{y}_{*,k} \sim P(\mathrm{y}_{*,k} \mid v'_k, h_k, u_k)$ using an *exact* sampler.
3. Accept v'_k with probability $\min\{1, r(v_k, v'_k, \mathrm{y}_{*,k} \mid \mathrm{z}_{*,k})\}$, where

$$r(v_k, v'_k, \mathrm{y}_{*,k} \mid \mathrm{z}_{*,k}) = \frac{P(v'_k)q(v_k \mid v'_k, \mathrm{z}_{*,k})}{P(v_k)q(v'_k \mid v_k, \mathrm{z}_{*,k})} \times \frac{P(\mathrm{z}_{*,k} \mid v'_k, h_k, u_k)P(\mathrm{y}_{*,k} \mid v_k, h_k, u_k)}{P(\mathrm{z}_{*,k} \mid v_k, h_k, u_k)P(\mathrm{y}_{*,k} \mid v'_k, h_k, u_k)}. \quad \text{(S.2)}$$

A problem lies in the second step. The exact sampling of $\mathrm{y}_{*,k}$ is as difficult as the original problem. The double MH sampler approximates it with a Gibbs sampling scan of $z_{l,k}$'s starting from the current $\mathrm{z}_{*,k}$. At each step of the Gibbs sampling scan, $z_{l,k}$ is updated according to $P(z_{l,k} \mid \mathrm{z}_{-l,k}, v'_k, h_k, u_k)$.

We construct the proposal distributions $q(v'_k \mid v_k, \mathrm{z}_{*,k})$ and $q(h'_k \mid h_k, \mathrm{z}_{*,k})$ using a log-normal distribution, and $q(u'_k \mid u_k, \mathrm{z}_{*,k})$ using a Gaussian distribution.

S.4 Hamiltonian Monte Carlo (HMC)

HMC (Neal, 2011) is a Markov chain Monte Carlo method for drawing samples from a probability density distribution. Unlike Metropolis-Hastings, it exploits gradient information to propose a new state, which can be distant from the current state. If no numerical error is involved, the new state proposed by HMC is accepted with probability 1.

HMC has a connection to Hamiltonian dynamics and the physical analogy is useful for gaining an intuition. In HMC, the variable to be sampled, $q \in \mathbb{R}^M$, is seen as a generalized coordinate of a system and is associated with a potential energy function $U(q) = -\log P(q)$, the negative logarithm of the (unnormalized) density function. The coordinate q is tied with an auxiliary momentum variable $p \in \mathbb{R}^M$ and a kinetic function $K(p)$. The momentum makes the object move. Since $H(q,p) = U(q) + K(p)$, the sum of the kinetic and potential energy, is constant with respect to time, the time evolution of the system is uniquely defined given an initial state (q_0, p_0). The trajectory is computed to get a state (q,p) at some time, and that q is the next sample we want.

Algorithm S.1 shows the pseudo-code, which is adopted from Neal (2011). The momentum variable is drawn from the Gaussian distribution (Line 2). The time evolution of the system is numerically simulated using the leapfrog method (Lines 4–11), where ϵ and S are parameters of the algorithm to be tuned. This is followed by a Metropolis step to correct for numerical errors (Lines 13–18).

Going back to the sampling of $\mathrm{w}_{k,*}$, we need $U(\mathrm{w}_{k,*}) = -\log P(\mathrm{w}_{k,*} \mid -)$ and its gradient $\nabla U(\mathrm{w}_{k,*})$ to run HMC. The unnormalized density function $P(\mathrm{w}_{k,*} \mid -)$ is the product of (1) the probability of generating $w_{k,m}$'s from the t-distribution and (2) the probability of generating $x_{l,i}$'s for each language with $z_{l,k} = 1$. Note that $U(\mathrm{w}_{k,*})$ is differentiable.

S.5 Details of Surface-DIA

A variant of the autologistic model, called Surface-DIA in the experiments, is based on the one proposed by Yamauchi and Murawaki (2016) and is also closely related to $P(Z \mid A)$ of SYNDIA. However, there are several differences.

Algorithm S.1 HMC($U, \nabla U, q_0$).

```
1:  $q \leftarrow q_0$
2:  $p_0 \sim \mathcal{N}(\mu = 0, \Sigma = I)$
3:  $p \leftarrow p_0$
4:  $p \leftarrow p - \epsilon \nabla U(q)/2$
5:  for $s \leftarrow 1, S$ do
6:      $q \leftarrow q + \epsilon p$
7:      if $s < S$ then
8:          $p \leftarrow p - \epsilon \nabla U(q)$
9:      end if
10: end for
11: $p \leftarrow p - \epsilon \nabla U(q)/2$
12: $p \leftarrow -p$
13: $r \sim \mathrm{Uniform}[0,1]$
14: if $\min[1, \exp(-U(q) + U(q_0) - K(p) + K(p_0))] > r$ then
15:     return $q$                                  ▷ accept
16: else
17:     return $q_0$                                ▷ reject
18: end if
```

S.5.1 Autologistic Model for Categorical Data

While the autologistic models within SYNDIA work on binary data, surface features are categorical. Recall that $\mathrm{x}_{*,i} = (x_{1,i}, \cdots, x_{L,i})$ is a sequence of feature values, where $x_{l,i}$ is the value of the l-th language for feature type i. $x_{l,i}$ takes one of F_i categorical values, and F_i differs according to feature types.

The counting functions for the vertical and horizontal factors, $V(\mathrm{x}_{*,i})$ and $H(\mathrm{x}_{*,i})$, are the same as those of the binary model. They return the number of pairs sharing the same value in the corresponding neighbor graph. However, we need to modify the counting function for the universal factor. In Surface-DIA, $U_j(\mathrm{x}_{*,i})$ returns the number of languages that take the value of j. Accordingly, the autologistic model has the following parameters for each surface feature i: vertical stability v_i, horizontal diffusibility h_i, and universality $\mathrm{u}_i = (u_{i,1}, \cdots, u_{i,F_i})$. The probability of $\mathrm{x}_{*,i}$, conditioned on v_i, h_i and u_i, can be expressed as:

$$P(\mathrm{x}_{*,i} | v_i, h_i, \mathrm{u}_i) \propto \exp\Bigg(v_i V(\mathrm{x}_{*,i}) + h_i H(\mathrm{x}_{*,i}) + \sum_j u_{i,j} U_j(\mathrm{x}_{*,i})\Bigg). \tag{S.3}$$

SYNDIA uses prior distributions to constrain v_k and h_k to always be positive. Since Surface-DIA is based on Yamauchi and Murawaki (2016) and

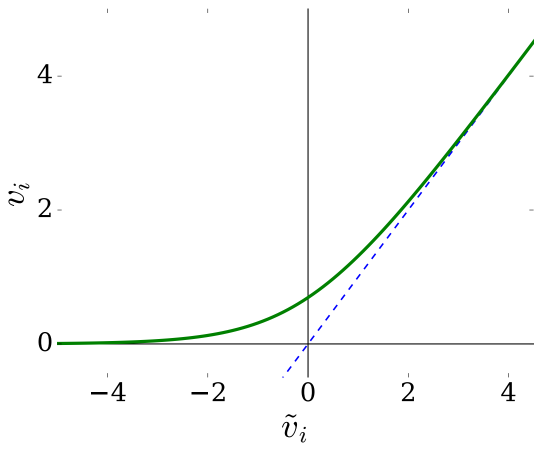

Figure S.1: Softplus function. The dashed line is $f(\tilde{v}_i) = \tilde{v}_i$.

does not use priors, we implement the constraints differently. Specifically, we reparameterize v_i and h_i with the softplus function:

$$v_i = \text{softplus}(\tilde{v}_i) = \log(1 + \exp(\tilde{v}_i)). \quad \text{(S.4)}$$

Figure S.1, which plots the softplus function, indicates a positive range.

S.5.2 Maximum Likelihood Estimation

While we perform posterior inference for SYN-DIA, Surface-DIA employs maximum likelihood estimation as it was done by Yamauchi and Murawaki (2016).

Let $\mathrm{x}_{*,i}$ be decomposed into an observed portion $\mathrm{x}^{\text{obs}}_{*,i}$ and the remaining missing portion be $\mathrm{x}^{\text{mis}}_{*,i}$. Marginalizing out $\mathrm{x}^{\text{mis}}_{*,i}$, we obtain the log-likelihood to be maximized:

$$\mathcal{L}^{\text{mis}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i}) = \log \sum_{\mathrm{x}^{\text{mis}\prime}_{*,i}} P(\mathrm{x}^{\text{obs}}_{*,i} \cup \mathrm{x}^{\text{mis}\prime}_{*,i} \mid v_i, h_i, \mathrm{u}_i).$$

To maximize the objective, we perform gradient-based training. A simple gradient ascent algorithm would iteratively update v_i as follows:

$$v_i \leftarrow v_i + \eta_t \frac{\partial \mathcal{L}^{\text{mis}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i})}{\partial v_i},$$

where η_t is a learning rate that decays in relation to time t. h_i and u_i are updated in similar ways.

The derivative of the log-likelihood function with respect to v_i is:

$$\begin{aligned}&\frac{\partial \mathcal{L}^{\text{mis}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i})}{\partial v_i} \\ &= \mathrm{E}_{\mathrm{x}^{\text{mis}\prime}_{*,i} \sim P(\mathrm{x}^{\text{mis}\prime}_{*,i} | \mathrm{x}^{\text{obs}}_{*,i}, v_i, h_i, \mathrm{u}_i)} \left[V(\mathrm{x}^{\text{obs}}_{*,i} \cup \mathrm{x}^{\text{mis}\prime}_{*,i}) \right] \\ &\quad - \mathrm{E}_{\mathrm{x}'_{*,i} \sim P(\mathrm{x}'_{*,i} | v_i, h_i, \mathrm{u}_i)} \left[V(\mathrm{x}'_{*,i}) \right].\end{aligned}$$

Following Yamauchi and Murawaki (2016), we approximate the expectations using Gibbs sampling.

With softplus reparameterization, we need to compute the derivative of the log-likelihood function with respect to $\tilde{v}_i$. Applying the chain rule of differentiation, we obtain

$$\begin{aligned}&\frac{\partial \mathcal{L}^{\text{mis}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i})}{\partial \tilde{v}_i} \\ &= \frac{\partial \mathcal{L}^{\text{mis}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i})}{\partial v_i} \frac{\partial v_i}{\partial \tilde{v}_i} \\ &= \frac{\partial \mathcal{L}^{\text{mis}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i})}{\partial v_i} \frac{1}{1 + \exp(-\tilde{v}_i)}.\end{aligned}$$

Once we have estimated $\tilde{v}_i$ (and $\tilde{h}_i$), we can use Eq. (S.4) to compute v_i (and h_i) as the final output.

$u_{i,j}$ is initialized with the log-probability of assuming the value j in $\mathrm{x}^{\text{obs}}_{*,i}$. We also add L2 regularization to impose a penalty on u_i that is very different from its initial value $\mathrm{u}^{(0)}_i$:

$$\begin{aligned}&\mathcal{L}^{\text{mis}-\text{L2}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i}) \\ &\quad = \mathcal{L}^{\text{mis}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i}) - \frac{\lambda}{2} \|\mathrm{u}_i - \mathrm{u}^{(0)}_i\|^2_2,\end{aligned}$$

where $\lambda > 0$ controls the strength of the penalty. The derivative of the log-likelihood function with respect to $u_{i,j}$ is:

$$\begin{aligned}&\frac{\partial \mathcal{L}^{\text{mis}-\text{L2}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i})}{\partial u_{i,j}} \\ &= \frac{\partial \mathcal{L}^{\text{mis}}(v_i, h_i, \mathrm{u}_i;\ \mathrm{x}^{\text{obs}}_{*,i})}{\partial u_{i,j}} - \lambda(u_{i,j} - u^{(0)}_{i,j}).\end{aligned}$$

S.5.3 Experimental Settings

For each surface feature i, we ran 200 iterations for estimating v_i, h_i and u_i. After that, we fixed the parameters and sampled $\mathrm{x}^{\text{mis}}_{*,i}$ as follows. After 100 burn-in iterations, we ran 2,495 iterations and collected samples with the interval of 5 iterations. For each language, we chose the most frequent feature value among the collected samples as the final output.

We initialized $\tilde{v}_i$ and $\tilde{h}_i$ with -5, which corresponded to $v_i = h_i \simeq 0.0067$. For L2 regularization, we set $\lambda = 50$. Instead of the simple stochastic gradient ascent algorithm, Adam (Kingma and Ba, 2015) was used with the following hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

Note that while Yamauchi and Murawaki (2016) partitioned each $\mathrm{x}^{\mathrm{obs}}_{*,i}$ into 10 equal sized subsets, we performed the 10-fold cross-validation at the level of $X^{\mathrm{obs}} = \mathrm{x}^{\mathrm{obs}}_{*,1} \cup \cdots \cup \mathrm{x}^{\mathrm{obs}}_{*,N}$ in order to allow comparison with other models. The slight unevenness did not seem to have made a notable impact on the overall performance.

S.6 Details of Syn

The baseline model Syn is a simplified version of SynDia. It is created by removing v_k and h_k from SynDia. With this modification, $P(z_{l,k} \mid \mathrm{z}_{-l,k}, v_k, h_k, a_k)$ is reduced to

$$P(z_{l,k} = 1 \mid u_k) = \frac{\exp(u_k)}{\exp(u_k) + 1},$$

where $u_k \sim \mathcal{N}(0, \sigma^2)$ as before. Note that if $u_k = 0$, then $P(z_{l,k} = 1 \mid u_k) = P(z_{l,k} = 0 \mid u_k) = 0.5$.

The experimental settings for Syn were basically the same as those of SynDia. The difference was that since Syn lacked v_k and h_k, we had to estimate them for each induced parameter k *after* the posterior inference.

S.6.1 Comparison with the IBP

Syn is comparable to the IBP-based $P(Z|A)$, which we discussed in Section S.1. In both models, $z_{l,k}$'s are conditionally independent given μ_k for the IBP and u_k for Syn.

In the preliminary experiments, Syn slightly outperformed the IBP in terms of missing value imputation. We conjecture that even with simulated annealing, the IBP's preference to sparse Z remained somewhat harmful.

S.6.2 Autologistic Model in the Pipeline

To estimate v_k and h_k for the parameters of Syn, we first collect samples of $z_{l,k}$. We then construct $P(\mathrm{z}_{*,k}; \Phi)$, the empirical probability according to the collected samples. It reflects the model's uncertainty about $z_{l,k}$.

The autologistic model we use is the one introduced in Section S.5.1. All we have to do is to substitute $\mathrm{x}_{*,i}$, v_i, h_i and u_i with $\mathrm{z}_{*,k}$, v_k, h_k and u_k, respectively.

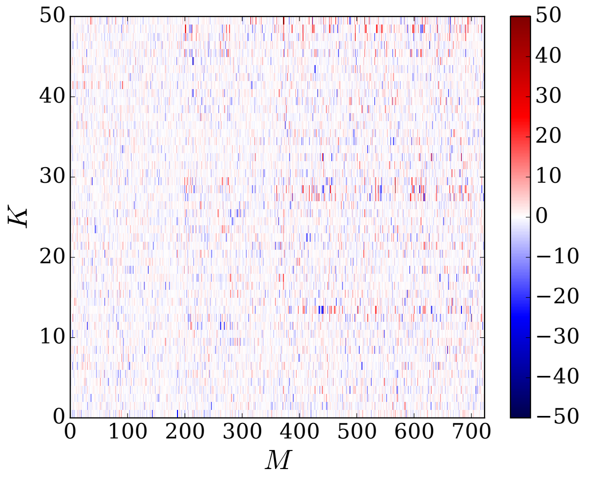

Figure S.2: Weight matrix W of SynDia ($K = 50$). Each row represents a parameter. Parameters are sorted by u_k in decreasing order.

However, we have to devise a new objective function because the data can no longer be decomposed into observed and missing portions. Instead, $z_{l,k}$'s are distributed according to $P(\mathrm{z}_{*,k}; \Phi)$. Now, the objective function for parameter k is

$$L^{\mathrm{P}}(v_k, h_k, \mathrm{u}_k; \Phi) = \mathrm{E}_{P(\mathrm{z}_{*,k};\Phi)}[\log P(\mathrm{z}_{*,k} | v_k, h_k, \mathrm{u}_k)].$$

As before, we perform gradient-based training to maximize the objective and use Gibbs sampling to approximate expectations. Parameter settings were the same as those described in Section S.5.3.

References

Mark C. Baker. 2002. *The Atoms of Language: The Mind's Hidden Rules of Grammar*. Basic Books.

Gabriel Doyle, Klinton Bicknell, and Roger Levy. 2014. Nonparametric learning of phonological constraints in optimality theory. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1103.

Dilan Görür, Frank Jäkel, and Carl Edward Rasmussen. 2006. A choice model with infinitely many latent features. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 361–368.

Thomas L. Griffiths and Zoubin Ghahramani. 2011. The Indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12:1185–1224.

Sonia Jain and Radford M. Neal. 2007. Splitting and merging components of a nonconjugate Dirichlet process mixture model. *Bayesian Analysis*, 2(3):445–472.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*.

David Knowles and Zoubin Ghahramani. 2007. Infinite sparse factor analysis and infinite independent components analysis. In *Proceedings of the 7th International Conference on Independent Component Analysis and Signal Separation*, pages 381–388.

Faming Liang. 2010. A double Metropolis–Hastings sampler for spatial models with intractable normalizing constants. *Journal of Statistical Computation and Simulation*, 80(9):1007–1022.

Edward Meeds, Zoubin Ghahramani, Radford Neal, and Sam Roweis. 2007. Modeling dyadic data with binary latent factors. In *NIPS*, volume 19, pages 977–984.

Jesper Møller, Anthony N. Pettitt, R. Reeves, and Kasper K. Berthelsen. 2006. An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, 93(2):451–458.

Iain Murray, Zoubin Ghahramani, and David J. C. MacKay. 2006. MCMC for doubly-intractable distributions. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 359–366.

Radford M. Neal. 2000. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Radford M. Neal. 2011. MCMC using Hamiltonian dynamics. In Steve Brooks, Andrew Gelman, Galin L. Jones, and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. CRC Press.

Yee Whye Teh. 2006. A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, National University of Singapore.

Yee Whye Teh and Dilan Görür. 2009. Indian buffet processes with power-law behavior. In *Advances in Neural Information Processing Systems*, pages 1838–1846.

Yee Whye Teh, Dilan Görür, and Zoubin Ghahramani. 2007. Stick-breaking construction for the Indian buffet process. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, pages 556–563.

Kenji Yamauchi and Yugo Murawaki. 2016. Contrasting vertical and horizontal transmission of typological features. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 836–846.