Getting It Right for the IPTM

Bomin Kim[1], Aaron Schein[3], Bruce Desmarais[1], and Hanna Wallach[2,3]

[1]Pennsylvania State University
[2]Microsoft Research NYC
[3]University of Massachusetts Amherst

November 11, 2017

Abstract

Software development is integral to the objective of applying the IPTM —the interaction-partitioned topic model— to real world data. Code review is a valuable process in any research computing context, and the prevalence of software bugs in statistical software is well documented (e.g., Altman et al., 2004; McCullough, 2009). With highly complex models such as the IPTM, there are many ways in which software bugs can be introduced and go unnoticed. As such, we present a joint analysis of the integrity of our generative model, sampling equations, and software implementation for the IPTM.

1 Getting It Right (GiR) Test

Geweke (2004) introduced the "Getting it Right" (GiR) test—a joint distribution test of posterior simulators which can detect errors in sampling equations as well as coding errors. The test involves comparing the distributions of variables simulated from two joint distribution samplers, which we call "forward" and "backward" samples. The forward sampler draws unobservable variables from the prior and then generates the observable data conditional on the unobservables. The backward sampler alternates between the inference and an observables simulator, by running the inference code on observable data to obtain posterior estimates of the unobervable variables and then re-generating the observables given the inferred unobservables. The backward sampler is initialized by running an iteration of inference on observables drawn directly from the prior. Since the only information on which both the forward and backward samplers are based is the prior, if the sampling equations are correct and the code is implemented without bugs, each variable should have the same distribution in the forward and backward samples. In other words, if there are no mistakes, then the forward samples and the backward samples should be indistinguishable.

1.1 IPTM: Forward Generating Process

In order to draw forward samples, we first draw all unobservable variables, *e.g.* $\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{l}, \boldsymbol{b}, \boldsymbol{\eta}, \delta$, from their respecitve prior distributions and then generates the observable data, *e.g.* $\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{t}$, conditional on the generated latent variables. The forward generating process of the IPTM exactly follows Section 2 in the main paper, summarized in Algorithm 1 below.

Algorithm 1 Forward Generating Process

Input:
1. Number of classes, *e.g.* D (documents), V (words), K (topics), $\{N_d\}_{d=1}^{D}$ (tokens), A (actors), and C (interaction patterns)
2. Timestamp distribution and link function, *e.g.* lognormal with identity link
3. Hyperparameters, *e.g.* $\alpha, \beta, \boldsymbol{m}, \boldsymbol{\mu}_b, \Sigma_b, \boldsymbol{\mu}_\eta, \Sigma_\eta, \mu_\delta, \sigma_\delta^2$ and (a_τ, b_τ) if σ_τ^2 needed.

Unobservables from prior distributions

- For $k = 1, \ldots, K$
  - $\boldsymbol{\phi}_k \sim \text{Dirichlet}\Big(\beta, (\frac{1}{V}, \ldots, \frac{1}{V})\Big)$
  - $l_k \sim \text{Uniform}(1, C)$
- For $d = 1, \ldots, D$,
  - $\boldsymbol{\theta}_d \sim \text{Dirichlet}\Big(\alpha, (m_1, \ldots, m_K)\Big)$
- For $c = 1, \ldots, C$
  - $\boldsymbol{b}_c \sim \text{Normal}(\boldsymbol{\mu}_b, \Sigma_b)$
  - $\boldsymbol{\eta}_c \sim \text{Normal}(\boldsymbol{\mu}_\eta, \Sigma_\eta)$
- $\delta \sim \text{Normal}(\mu_\delta, \sigma_\delta^2)$
- $\sigma_\tau^2 \sim \text{Inverse-Gamma}(a_\tau, b_\tau)$

Observables from generative process

for *d=1 to D* **do**
  for *n=1 to* N_d **do**
    draw $z_{dn} \sim \text{Multinomial}(\boldsymbol{\theta}_d))$
    draw $w_{dn} \sim \text{Multinomial}(\phi_{z_{dn}})$
  end
  for *c=1 to C* **do**
    set $\pi_{dc} = \frac{\sum_{k:l_k=c} N_{dk}}{N_d}$
  end
  for *i=1 to A* **do**
    for *j = 1 to A* **do**
      calculate $\boldsymbol{\nu}_{idjc} = \{\nu_{idjc}\}_{c=1}^{C}$, where $\nu_{idjc} = \exp\Big(\boldsymbol{b}_c{}^\top \boldsymbol{x}_{idjc}\Big)$
      calculate $\lambda_{idj} = \sum_{c=1}^{C} \pi_{dc}\, \nu_{idjc}$,
    end
    draw $\boldsymbol{u}_{id} \sim \text{Gibbs}(\delta, \boldsymbol{\lambda}_{id})$
    calculate $\boldsymbol{\xi}_{idc} = \{\xi_{idc}\}_{c=1}^{C}$, where $\xi_{idc} = \boldsymbol{\eta}_c^\top \text{GeomMean}(\{\boldsymbol{y}_{idjc}\}_{j:u_{idj}=1})$
    calculate $\mu_{id} = \sum_{c=1}^{C} \pi_{dc} g^{-1}(\xi_{idc})$
    draw $\tau_{id} \sim \phi_\tau(\mu_{id}, \sigma_\tau^2)$, where ϕ_τ is the pdf of specified timestamp distribution
  end
  set $a_d = \text{argmin}_i(\tau_{id})$, $\boldsymbol{r}_d = \boldsymbol{u}_{a_d d}$, and $t_d = t_{d-1} + \tau_{a_d d}$.
end

1.2 IPTM: Backward Generating Process

Next, we take one of these forward samples and, given this sample, draw a sample of the latent variables from their posterior distribution using our inference code. We then use the final step of the generative process to draw a new sample of the data given this sample of the latent variables. Finally, we repeat the last two steps many times to obtain a set of backward samples.

For backward sampling, we let N_{vk} be the number of tokens of word-type v that are currently assigned to topic k. Also let N_k be the total number of tokens currently assigned to topic k. Word-assignments are implemented via collapsed Gibbs sampling (Griffiths, 2002), while the generation of tie data $(a_d, \boldsymbol{r}_d, t_d)$ directly follows the generating process, same as the forward generating process. This "backward" version of the generative process is illustrated in Algorithm 2 below.

Algorithm 2 Backward Generating Process

Input:
1. One set of forward sample
2. Inference setting, *e.g.* number of outer iterations, length of Metropolis-Hastings (M-H) chains
3. Timestamp distribution and link function, *e.g.* lognormal with identity link

Inference on one forward sample
for *o = 1 to O* **do**
  Run inference code according to Algorithm 1 in IPTM paper (Appendix B.5)
  (NOTE: ensure full convergence on all latent variables)
end

Unobservables from inference
- For $k = 1, \ldots, K$
  - obtain the last of l_k
- For $d = 1, \ldots, D$,
  - obtain the last sample of $\boldsymbol{z}_d$
- For $c = 1, \ldots, C$
  - obtain the posterior mean of $\boldsymbol{b}_c$
  - obtain the posterior mean of $\boldsymbol{\eta}_c$
- obtain the posterior mean of δ
- obtain the posterior mean of σ_τ^2,

where the posterior means are calculated from MCMC chains via M-H.

Observables from generative process
set all $N_{vk} = 0$ and $N_k = 0$
for *d=1 to D* **do**
  for *n=1 to* N_d **do**
    for *v=1 to V* **do**
      token-word-type-distribution$_{dn}[v] = \frac{N_{z_{dn}v} + \beta/V}{N_{z_{dn}} + \beta}$
    end
    draw $w_{dn} \sim$ (token-word-type-distribution$_{dn}$)
    $N_{w_{dn}, z_{dn}} += 1$
    $N_{z_{dn}} += 1$
  end
  for *c=1 to C* **do**
    set $\pi_{dc} = \frac{\sum_{k:l_k=c} N_{dk}}{N_d}$
  end
  for *i=1 to A* **do**
    for *j = 1 to A* **do**
      calculate $\boldsymbol{\nu}_{idjc} = \{\nu_{idjc}\}_{c=1}^{C}$, where $\nu_{idjc} = \exp\left(\boldsymbol{b}_c{}^\top \boldsymbol{x}_{idjc}\right)$
      calculate $\lambda_{idj} = \sum_{c=1}^{C} \pi_{dc}\, \nu_{idjc}$,
    end
    draw $\boldsymbol{u}_{id} \sim \text{Gibbs}(\delta, \boldsymbol{\lambda}_{id})$
    calculate $\boldsymbol{\xi}_{idc} = \{\xi_{idc}\}_{c=1}^{C}$, where $\xi_{idc} = \boldsymbol{\eta}_c^\top \text{GeomMean}(\{\boldsymbol{y}_{idjc}\}_{j:u_{idj}=1})$
    calculate $\mu_{id} = \sum_{c=1}^{C} \pi_{dc} g^{-1}(\xi_{idc})$
    draw $\tau_{id} \sim \phi_\tau(\mu_{id}, \sigma_\tau^2)$, where ϕ_τ is the pdf of specified timestamp distribution
  end
  set $a_d = \text{argmin}_i(\tau_{id})$, $\boldsymbol{r}_d = \boldsymbol{u}_{a_d d}$, and $t_d = t_{d-1} + \tau_{a_d d}$.
end

2 Statistics to Use

To compare the forward and backward samples, we need some discrepancy functions that measure various properties of generated data. For each forward and backward sample that consists of D number of documents, we save these statistics:

1. Mean of network effect parameters $(\boldsymbol{b}_p^{(1)}, ..., \boldsymbol{b}_p^{(C)})$ for every $p = 1, ..., P$,
2. Network statistic 'send' calculated for the last D^{th} document for every $l = 1, ..., 3$
3. δ value used to generate the samples
4. Mean of the observed recipient size $\|J_o^{(d)}\|_1$ across $d = 1, ..., D$,
5. Mean of time-increments $t^{(d)} - t^{(d-1)}$ across $d = 1, ..., D$,
6. Mean topic-interaction pattern assignment c_k across $k = 1, ..., K$,
7. Number of tokens in topics assigned to each interaction pattern $c = 1, ..., C$,
8. Number of tokens assigned to each topic $k = 1, ..., K$,
9. Number of tokens assigned to each unique word type $w = 1, ..., W$,
10. Time-increment parameter $\boldsymbol{\eta}$ used to generate the samples, for every $q = 1, .., Q$
11. Log-normal variance parameter σ_T^2 used to generate the samples

3 Results

3.1 Schein Test

3.2 GiR Test

These are our results. All the blue dots lie on the 45-degree red line, so our derivations and code pass the test. I do want to note that they didn't originally pass the test. In fact, when Bruce presented our work at PolMeth they were still failing! But we've now tracked down and fixed the mistakes that were responsible.

To keep the computational burden of re-running thousands of rounds of inference manageable, we run GiR using a relatively small artificial sample, consisting of 5 documents, 4 tokens per document, 4 actors, 5 unique word types, 2 interaction patterns, and 4 topics per each forward or backward samples. For detailed settings including the prior specifications, see Appendix A.2. We generated 10^5 sets of forward and backward samples, and then calculated 1,000 quantiles for each of the network effect parameters, and 50 quantiles for the rest of the statistics. We also calculated t-test and Mann-Whitney test p-values in order to test for differences in the distributions generated in the forward and backward samples. Before we calculated these statistics, we thinned our samples by taking every 9th sample starting at the 10,000th sample for a resulting sample size of 10,000, in order to reduce the autocorrelation in the Markov chains. In each case, if we observe a large p-value, this gives us evidence that the distributions generated under forward and backward sampling have the same locations. We depict the GiR results using probability-probability (PP) plots. To compare two samples with a PP-plot we calculate the empirical quantile in each sample of a set of values observed across the two samples, then plot the sets of quantiles in the two samples against each other. If the two samples are from equivalent distributions, the quantiles should line up on a line with zero y-intercept, and unit slope (i.e., a 45-degree line). The GiR test results are depicted in Figure 2, which show that we pass the test on every statistic.

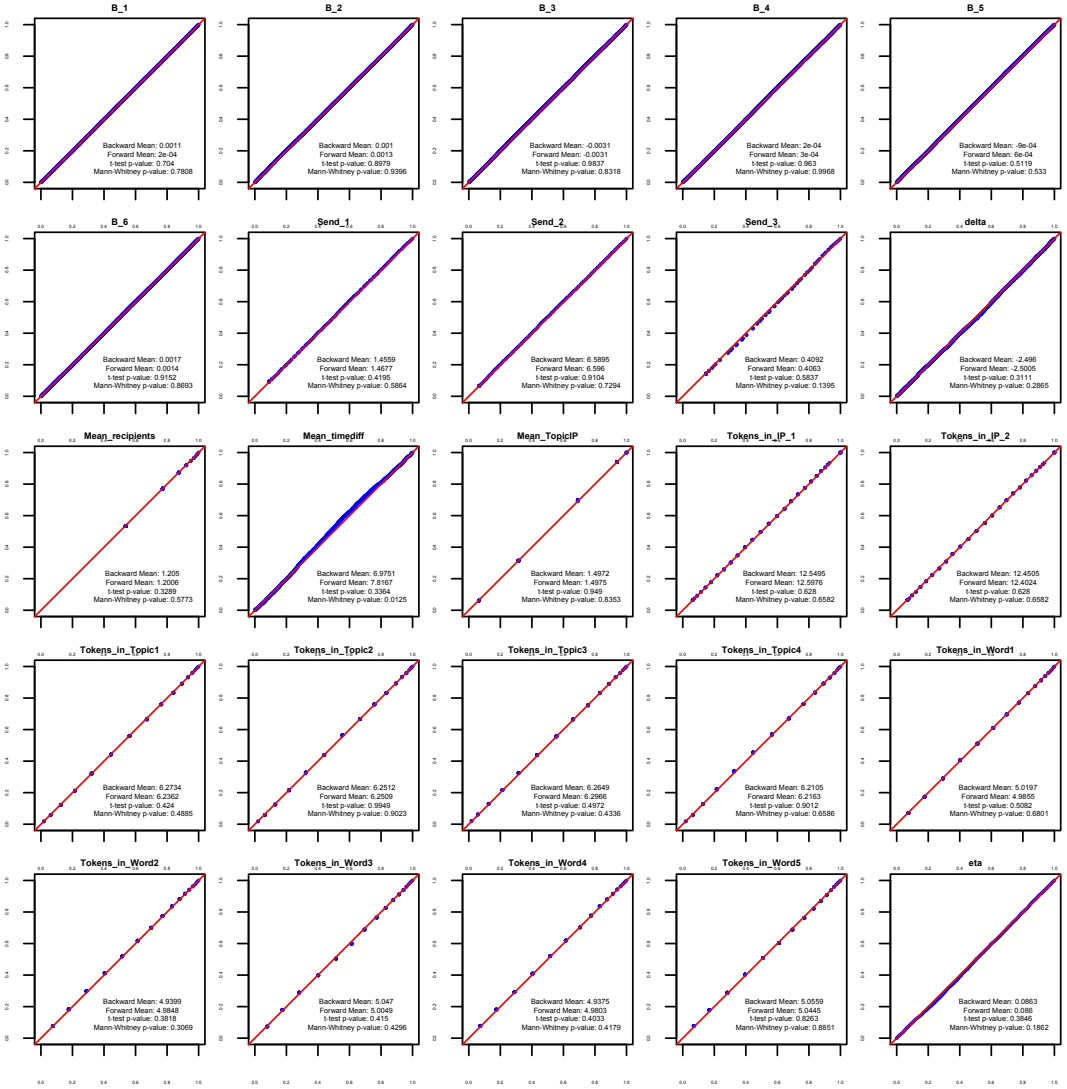

Figure 1: Probability-Probability plot for the 25 GiR test statistics.

4 Conclusion

"Getting it Right" test is very simple to implement, yet very powerful because it amplifies subtle misktakes, either in the sampling equations or code. We hope that all researchers should be using GiR whenever they derive and implement Markov Chain Monte Carlo (MCMC) inference for a Bayesian latent variable model.

References

Altman, M., Gill, J., and McDonald, M. P. (2004). *Numerical issues in statistical computing for the social scientist*, volume 508. John Wiley & Sons.

Geweke, J. (2004). Getting it right: Joint distribution tests of posterior simulators. *Journal of the American Statistical Association*, 99(467):799–804.

Griffiths, T. (2002). Gibbs sampling in the generative model of latent dirichlet allocation.

McCullough, B. D. (2009). The accuracy of econometric software. *Handbook of computational econometrics*, pages 55–79.

Appendix

A Details on Getting It Right Test

A.1 Intitialization of History $\boldsymbol{x}_t^{(c)}$

Considering that our network statistics $\boldsymbol{x}_t^{(c)}$ are generated as a function of the network history, it is necessary to use the same initial value of $\boldsymbol{x}_t^{(c)}$ across the forward and backward samples. If not, when we generate fixed number of documents, we cannot guarantee the same number of documents used for the inference, since only the documents with its timestamp greater than 384 hours are used in the inference. In the extreme cases, we may end up with two types of failure:

1. Zero document generated after 384 hours (i.e. $t^{(10)} < 384$), making no documents to be used for inference,
2. Zero document generated before 384 hours (i.e. $t^{(1)} > 384$), making the estimate of $\mathcal{B}$ totally biased since $\forall \boldsymbol{x}_t^{(c)}(i,j) = 0$.

Therefore, we fix the initial state of $\boldsymbol{x}_t^{(c)}$ over the entire GiR process. Specifically, we fix some baseline documents where the timestamps are all smaller than 384 and use as an input for forward sampling, backward sampling, and the inference. Then, in the forward and backward generative process, we set the starting point of the timestamp as $t^{(0)} = 384$ and generate fixed number of documents given the initial $\boldsymbol{x}_{t^{(0)}=384}^{(c)}$ so that we can acheive consistency in the generated number of documents with $t^{(d)} > 384$.

A.2 GiR Implementation Details

While we tried a number of different parameter combinations in the course of testing, we outline our standard setup. We selected the following parameter values:

- D (number of documents) = 5
- $N^{(d)}$ (tokens per document) = 4
- A (number of actors)= 4
- W (unique word types) = 5
- C (number of interaction patterns) = 2
- K (number of topics) = 4
- α (Dirichlet concentration prior) = 2
- $\boldsymbol{m}$ (Dirichlet base prior) = $\boldsymbol{u}$
- β (Dirichlet concentration prior)= 2
- $\boldsymbol{n}$ (Dirichlet base prior) = $\boldsymbol{u}$
- netstat = "intercept" and "dyadic"
- prior for $\boldsymbol{b}^{(c)}$: $\mu_{\boldsymbol{b}^{(c)}} = (-3, \mathbf{0}_6), \Sigma_{\boldsymbol{b}^{(c)}} = 0.05 \times I_7$
- prior for δ: $\mu_\delta = 2.5$, $\sigma_\delta^2 = 0.0001$
- prior for η: $\mu_\eta = 2.5$, $\sigma_\eta^2 = 0.0001$
- I (outer iteration) = 5
- n_1 (hyperparamter optimization) = 0
- n_2 (M-H sampling iteration of $\mathcal{B}$) = 5500
- burn (M-H sampling burn-in of $\mathcal{B}$)= 500
- thin (M-H sampling thinning of $\mathcal{B}$)= 5
- σ_{Q1}^2 (proposal variance for $\mathcal{B}$) = 0.1
- n_3 (M-H sampling iteration of δ) = 500
- σ_{Q2}^2 (proposal variance for δ) = 0.0002