Sampling From a Discrete Posterior

Bar Weinstein

December 26, 2024

1 Setup

We have posterior

$$\begin{aligned} p(\eta, \theta, \gamma, A^* | Y, Z, X, A) \propto & \, p(Y | Z, A^*, X, \eta) p(\eta) \\ & \times p(A | A^*, X, \gamma) p(\gamma) \\ & \times p(A^* | X, \theta) p(\theta), \end{aligned} \tag{1}$$

where η, θ, γ are continuous and A^* is discrete with large dimension $O(N^2)$.

First approach is to use the cut-posterior

$$p_{cut}(\eta, \theta, \gamma, A^* | Y, Z, X, A) \propto p(\eta | Y, Z, A^*, X) p(\theta, \gamma, A^* | A, X), \tag{2}$$

and sample from it by first sample θ, γ, then generate A^* samples, and finally sample η either via plug-in or multi-stage sampling. This approach is fine but other than plug-in sampling it takes a while. We can maybe do better!

Gibbs sampling requires sampling from the full conditional of each. Write $O = (Y, Z, X, A)$ for the observed data. Then the full conditionals are

$$\begin{aligned} \gamma &\sim p(\gamma | O, \eta, \theta, A^*) \propto p(A | A^*, X, \gamma) p(\gamma) \\ \theta &\sim p(\theta | O, \eta, \gamma, A^*) \propto p(A^* | X, \theta) p(\theta) \\ \eta &\sim p(\eta | O, \theta, \gamma, A^*) \propto p(Y | Z, A^*, X, \eta) p(\eta) \\ A^* &\sim p(A^* | O, \eta, \theta, \gamma) \propto p(Y | Z, A^*, X, \eta) p(A | A^*, X, \gamma) p(A^* | X, \theta). \end{aligned} \tag{3}$$

Sampling γ, θ, η is easy, but sampling A^* is hard. The continuous paramteters can be updated with gradient-based methods such as MALA or HMC. I'll now present methods for sampling A^*.

2 Methods for Discrete Sampling

Assume that $\mathcal{A}^*$ is the space of all possible A^*. Naively, we can draw A^* by sampling all edges independently. However, A^* may have global impact on Y so flipping one edge A^*_{ij} can impact the likelihood of Y_k where $k \neq i, j$. That will complicate the naive A^* updates, e.g., via MH.

Write the log conditional posterior of A^* as

$$\begin{aligned} f(A^*) &= \log p(A^* | O, \eta, \theta, \gamma) \\ &\propto \log p(Y | Z, A^*, X, \eta) + \log p(A | A^*, X, \gamma) + \log p(A^* | X, \theta). \end{aligned} \tag{4}$$

1. **The Hamming Ball Sampler.**[1] Variant of slice sampling for discrete variables. Idea is to have an auxilary variable that constraint the sampling space to a subset of $\mathcal{A}^*$. At each iteration we create a "slice" (the auxilary variable), sample A^* from this slice subset, etc. Example of application is to restrict the number of changes of A^* proposal in each iteration and sample from that the Hamming ball of radius d. Effectively, it is local updates constraints to the number of new changes.

2. **Informed Proposals**. Basic idea is that in updating A^* we propose a new value from a kernel $q(A^{*'}|A^*)$. In fact, we propose each edge seperately so it can be written as $q(A^{*'}|A^*) = \sum_e q(A^{*'}|A^*, e)q(e)$, that is, sample edges with $q(e)$ and then "flip" the edge with $q(A^{*'}|A^*, e) = I(A^{*'}_{ij} = A^*_{ij})$, $\forall (i,j) \neq e$. Key idea here is what edges e we select to flip. Naive random-walk just select these edges uniformly, but we can do better. That is the main idea behind Zanella (2020)[2] paper on Informed Proposals.

   Assume for now that we only "flip" one edge at a time (can be extended for block or sequential updates in each iteration). That is, updates are limited to a local neighbor of Hemming distance 1. One option is to inform the edge selection by its impact on $f(A^*)$ values, i.e., on the likelihood ratio. As before, write $A^{*'}$ as A^* where only edge e is "flipped" $(0 \to 1, 1 \to 0)$. Then the score, or log likelihood ratio, is $d(A^{*'}, A^*) = f(A^{*'}) - f(A^*)$. We can use this score to inform the edge selection by taking proposals $q(A^{*'}|A^*) \propto g(\exp(d(A^{*'}, A^*)))$ for function g that satisfy $g(a) = ag(1/a)$, e.g., $g(a) = \sqrt{a}, g(a) = \frac{a}{a+1}$. For $g(a) = \frac{a}{a+1}$ we can take the proposal as $q(A^{*'}|A^*) \propto Softmax(d(A^{*'}, A^*)/\tau)$, where τ is a "temprature" controlling the locality of the updates. By Zanella (2020), optimal τ is $\tau = 2$ in the binary case (so called "Barker choice"). The scores are effectively the tempered likelihood ratios. However, computing $d(A^{*'}, A^*)$ is expensive since we need to evaluate f twice for each possible edges, i.e., at least $O(N^2)$ per iteration. Grathwohl et al. (2021)[3] proposed to approximate this score for $g(a) = \frac{a}{a+1}$ by taking a gradient: $\widetilde{d}(A^{*'}, A^*) = (A^{*'}_e - A^*_e)\nabla f_{A^*_e}(A^*_e)$. That is, $\widetilde{d}(A^{*'}, A^*) \approx d(A^{*'}, A^*) = f(A^{*'}) - f(A^*)$, where the gradient of $f()$ is taken w.r.t. A^*_e as if A^*_e is continuous[4]. They have some results about this approximation. The main benefit is computional since computing $d(A^{*'}, A^*)$ requires $O(N^2)$ evaluations whereas $\widetilde{d}$ requires only $O(1)$. The proposed pseudo-algorithm for A^* updates is thus:

   Namely, compute scores for edge selection, sample edge with probabilities proportional to the scores, flip its edge, compute new scores, and accept/reject with MH. This can be extended for either K edges "flips" in a block or K sequential flips in each iteration. The cool thing is that don't really have to flip all edges in each iteration.

   Limitations. To compute the exact scores (log of likelihood ratio) $d(A^{*'}, A^*)$ we need to evalutate f twice for each edge flip, i.e., $O(N^2)$ evaluations. If A^* and A are edge indepdent then, evaluating $\log p(A|A^*, X, \gamma) + \log p(A^*|X, \theta)$ is easy since each flip only affects one edge. However, the outcome model can possibly depend on global network statistics, e.g., eigenvector centrality, so flipping one edge can impact

[1] `https://www.tandfonline.com/doi/full/10.1080/01621459.2016.1222288`

[2] `https://www.tandfonline.com/doi/full/10.1080/01621459.2019.1585255?src=recsys`

[3] `https://proceedings.mlr.press/v139/grathwohl21a.html`

[4] It can be performed automatically in torch via `torch.autograd.grad` or in JAX via `jax.grad`.

Algorithm 1 Gibbs With Gradients

Input: unnormalized log-prob $f(\cdot)$, current sample x
Compute $\tilde{d}(x)$ {Eq. 3 if binary, Eq. 4 if categorical.}
Compute $q(i|x) = \text{Categorical}\left(\text{Softmax}\left(\frac{\tilde{d}(x)}{2}\right)\right)$
Sample $i \sim q(i|x)$
$x' = \texttt{flipdim}(x, i)$
Compute $q(i|x') = \text{Categorical}\left(\text{Softmax}\left(\frac{\tilde{d}(x')}{2}\right)\right)$
Accept with probability:

$$\min\left(\exp(f(x') - f(x))\frac{q(i|x')}{q(i|x)}, 1\right)$$

the outcomes of many units! That will be computionally expensive to evaluate the scores using the outcomes model. One option is to neglect the outcome model in the scores computation, or just use them occasionally (e.g., in every L iterations). That resemble cut-posterior sampling, but not exactly as we sample everything together. Need to think if using the gradient with $\widetilde{d}$ will assist even with the outcome model. Probably depend on the 'grad' functions implementation.

Multiple updates. In Grathwohl et al. (2021) approach, in each iteration we "flip" one edge. It can be modified to sampling K edges from $q(i|A^*)$ and in the accept/reject step write accordingly the new network $A^{*'}$ with all the selected edges flipped, and instead of $q(i|A^*)$ write the product $\prod_k q(i_k|A^*)$.

Discrete MALA. Zhang et al. (2022)[5] proposed to use *discrete Metropolis-adjusted Langevin algorithm (DMALA)*. Their approach builds on Grathwohl et al. (2021). Their idea is that if we can factorize the discrete parameter space $\mathcal{A}^* = \prod_e \mathcal{A}_e^* = \{0,1\}^D$ with $D = N(N-1)/2$, then we can suggest an update for all edges in each iteration, instead of the single update in Grathwohl et al. (2021). Their methods begins with computing the gradient $\nabla f(A^*)$ as approximation for the difference (likelihood ratio) as before. Then we "flip" edge e with probability $\frac{\exp\left(\frac{1}{2}\nabla f(A^*)_e(A_e^{*'} - A_e^*) - \frac{1}{2\alpha}\right)}{\exp\left(\frac{1}{2}\nabla f(A^*)_e(A_e^{*'} - A_e^*) - \frac{1}{2\alpha}\right)+1}$ where α is the step size that need to be tuned. Namely, we compute once the gradient for all edges, and then propose update (flip) for each edge at once (simultaneously edge updates). We accept/reject the update with MH step with the log-probabilities f and the product of selection probabilities. In our case, if we include the outcome model in $f(A^*)$, then eventhough the space $\mathcal{A}^*$ is factorized, the posterior edges are not neccessarily independent. Therefore, updating all edges simultaneously might not be accurate. Need to see how this is work in practice. This DMALA appraoch was rigrourously analyzed by Sun et al. (2023)[6]. They showed how Discrete Langevian Monte Carlo is derived from continuous Lanegvian Monte Carlo using Markov Jump Process. Zhang et al. (2022) is a specific case of their approach.

[5] `https://proceedings.mlr.press/v162/zhang22t.html`
[6] `https://proceedings.mlr.press/v206/sun23f.html`

Algorithm 2 DULA and DMALA with Binary Variables.

given: Stepsize α.
loop
compute $P(\theta) = \frac{\exp(-\frac{1}{2}\nabla U(\theta) \odot (2\theta - 1) - \frac{1}{2\alpha})}{\exp(-\frac{1}{2}\nabla U(\theta) \odot (2\theta - 1) - \frac{1}{2\alpha}) + 1}$
sample $\mu \sim \text{Unif}(0, 1)^d$
$I \leftarrow \texttt{dim}(\mu \leq P(\theta))$
$\theta' \leftarrow \texttt{flipdim}(I)$
▷ Optionally, do the MH step
compute $q(\theta'|\theta) = \prod_i q_i(\theta'_i|\theta) = \prod_{i \in I} P(\theta)_i \cdot \prod_{i \notin I}(1 - P(\theta)_i)$
compute $P(\theta') = \frac{\exp(-\frac{1}{2}\nabla U(\theta') \odot (2\theta' - 1) - \frac{1}{2\alpha})}{\exp(-\frac{1}{2}\nabla U(\theta') \odot (2\theta' - 1) - \frac{1}{2\alpha}) + 1}$
compute $q(\theta|\theta') = \prod_i q_i(\theta_i|\theta') = \prod_{i \in I} P(\theta')_i \cdot \prod_{i \notin I}(1 - P(\theta')_i)$
set $\theta \leftarrow \theta'$ with probability

$$\min\left(1, \exp\left(U(\theta') - U(\theta)\right) \frac{q(\theta|\theta')}{q(\theta'|\theta)}\right)$$

end loop
output: samples $\{\theta_k\}$

3 Implementation

To implment any of of the methods based on Local Proposal, we need the following

1. Evaluate the log conditional posterior $f(A^*)$ given the observed data and the current values of the continuous parameters.

2. Compute the gradient $\nabla f(A^*)$ given the above. Both 1. and 2. can be achieved using automatic differentiation libraries, e.g., using `jax.value_and_grad`.

3. Assuming I use proposal with $g(a) = \frac{a}{a+1}$, need also to compute $-(2A^* - 1)\nabla f(A^*)$ in each iteration.

4. Consider single or multiple updates via GWG. Or use DMALA.

5. Combine Discrete updates of A^* with the continuous updates of, e.g., η, θ, γ. Continuous updates can be done using MALA or HMC. The combination is within Gibbs sampling. For example, HMC-within-Gibbs that iterate between discrete and continuous updates.