2nd Symposium on Advances in Approximate Bayesian Inference, 2019 1–10

Newtonian Monte Carlo: a second-order gradient method for speeding up MCMC.

Nimar S. Arora, Nazanin Khosravani Tehrani, Kinjal Divesh Shah, Michael Tingley, Yucen Lily Li, Narjes Torabi, David Noursi, Sepehr Akhavan Masouleh, Eric Lippert, Erik Meijer

{NIMARORA, NAZANINKT, KSHAH97, TINGLEY, YUCENLI, NTORABI, DCALIFORNIA, SEPEHRAKHAVAN, ERICLIPPERT, ERIKM }@FB.COM

Facebook Inc, 1 Hacker Way, Menlo Park CA 94025

Abstract

We present Newtonian Monte Carlo (NMC), a method to improve Markov Chain Monte Carlo (MCMC) convergence by analyzing the first and second order gradients of the target density to determine a suitable proposal density at each point. Existing first order gradient-based methods suffer from the problem of determining an appropriate step size. Too small a step size and it will take a large number of steps to converge, while a very large step size will cause it to overshoot the high density region. NMC is similar to the Newton-Raphson update in optimization where the second order gradient is used to automatically scale the step size in each dimension. However, our objective is not to find a maxima but instead to find a parameterized density that can best match the local curvature of the target density. This parameterized density is then used as a single-site Metropolis-Hastings proposal.

As a further improvement on first order methods, we show that random variables with constrained supports don't need to be transformed before taking a gradient step. NMC directly matches constrained random variables to a proposal density with the same support thus keeping the curvature of the target density intact.

We demonstrate the efficiency of NMC on a number of different domains. For statistical models where the prior is conjugate to the likelihood, our method recovers the posterior quite trivially in one step. However, we also show results on fairly large non-conjugate models, where NMC performs better than adaptive first order methods such as NUTS or other inexact scalable inference methods such as Stochastic Variational Inference or bootstrapping.

1. Introduction

Markov Chain Monte Carlo (MCMC) methods are often used to generate samples from an unnormalized probability density $\pi(\theta)$ that is easy to evaluate but hard to directly sample. Such densities arise quite often in Bayesian inference as the posterior of a generative model $p(\theta, Y)$ conditioned on some observations $Y = y$, where $\pi(\theta) = p(\theta, y)$. The typical setup is to select a *proposal* distribution $q(.|\theta)$ that proposes a move of the Markov chain to a new state $\theta^* \sim q(.|\theta)$. This Metropolis-Hastings acceptance rule is then used to accept or reject this move with probability: $\min\left[1, \frac{\pi(\theta^*)q(\theta|\theta^*)}{\pi(\theta)q(\theta^*|\theta)}\right]$.

When $\theta \in \mathbb{R}^k$, a common proposal density is the Gaussian distribution $\mathcal{N}(\theta, \epsilon^2 I_k)$ centered at θ with covariance $\epsilon^2 I_k$, where ϵ is the step size and I_k is the identity matrix defined over $\mathbb{R}^{k,k}$. This proposal forms the basis of the so-called Random Walk MCMC (RWM) first proposed in Metropolis et al. (1953).

In cases where the target density $\pi(\theta)$ is differentiable, an improvement over the basic RWM method is to propose a new value in the direction of the gradient, as follows:

$$q(.|\theta) = \mathcal{N}\left(\theta + \frac{\epsilon^2}{2}\nabla \log\{\pi(\theta)\}, \epsilon^2 I_k\right).$$

This method is known as Metropolis Adjusted Langevin Algorithm (MALA), and arises from an Euler approximation of a Langevin diffusion process (Robert and Tweedie, 1996). MALA has been shown to reduce the number of steps required for convergence to $O(n^{1/3})$ from $O(n)$ for RWM (Roberts and Rosenthal, 1998). An alternate approach, which also uses the gradient, is to do an L-step Euler approximation of Hamiltonian dynamics known as Hamiltonian Monte Carlo (Neal, 1993), although it was originally published under the name Hybrid Monte Carlo (Duane et al., 1987).

In HMC the number of steps, L, can be learned dynamically by the No-U-Turn Sampler (NUTS) algorithm (Hoffman and Gelman, 2014). However, in all three of the above algorithms – RWM, MALA, and HMC – there is an open problem of selecting the optimal step size. Normally, the step size is adaptively learned by targeting a desired acceptance rate. This has the unfortunate effect of picking the same step size for all the dimensions of θ, which forces the step size to accomodate the dimension with the smallest variance as pointed out in Girolami and Calderhead (2011). The same paper introduces alternate approaches, using Reimann manifold versions of MALA (MMALA) and HMC (RMHMC). They propose a Reimann manifold using the expected Fisher information matrix plus the negative Hessian of the log-prior as a metric tensor, $-E_{y|\theta}\left[\frac{\partial^2}{\partial\theta^2} log\{p(y,\theta)\}\right]$, and proceed to derive the Langevin diffusion equation and Hamiltonian dynamics in this manifold. The use of the above metric tensor does address the issue of differential scaling in each dimension. However, the method as presented requires analytic knowledge of the Fisher information matrix. This makes it difficult to design inference techniques in a generic way, and requires derivation on a per-model basis. A more practical approach involves using the negative Hessian of the log-probability as the metric tensor, $\frac{\partial^2}{\partial\theta^2} log\{p(y,\theta)\}$. However, this encounters the problem that this is not necessarily positive definite throughout the state space. An alternate approach for scaling the moves in each dimension is to use a preconditioning matrix M (Roberts and Stramer, 2002) in MALA, $q(.|\theta) = \mathcal{N}\left(\theta + \epsilon^2 M\nabla \log\{\pi(\theta)\}, \epsilon^2 M\right)$, also known as the mass matrix in HMC and NUTS, but it's unclear how to compute this.

An alternate approach is to approximately compute the Hessian (Zhang and Sutton, 2011) using ideas from quasi-Newton optimization methods such as L-BFGS (Nocedal and Wright, 2006). This approach and its stochastic variant (Simsekli et al., 2016) use a fixed window of previous samples of size M to approximate the Hessian. However, this makes the chain an order M Markov chain, which introduces considerable complexity in designing the transition kernel in addition to introducing a new parameter M. The key observation in our work is that for single-site methods we only need to compute the Hessian of one coordinate at a time, and this is usually tractable. The other key observation is that we don't need to always make a Gaussian proposer using the Hessian. In some cases, other densities which are less concentrated such as Cauchy are more appropriate. In general, the Hessian can be used for the purpose of matching the curvature of any parameterized density that best approximates the conditional posterior. This approach of curvature-matching to

an approximating density allows us to deal with constrained random variables without introducing a transformation such as in Stan (Carpenter et al., 2017).

In the rest of the paper, we will describe our approach to exploit the curvature of the target density, and show some results on multiple data sets.

2. Newtonian Monte Carlo

This paper introduces the Newtonian Monte Carlo (NMC) technique for sampling from a target distribution via a proposal distribution that incorporates curvature around the current sample location. We wish to choose a proposal distribution that uses second order gradient information in order to closely match the target density. Whereas related MCMC techniques discussed in Section 1 may utilize second order gradient information, those techniques typically use it only to adjust step size when simulating steps along the general direction of the target density's gradient.

Our proposed method involves matching the target density to a parameteric density that best explains the current state. We have a library of 2-parameter target densities F_i, and simple inference rules such that, given the first and second order gradients, we can solve the following two equations:

$$\nabla \log\{\pi(\theta)\} = \frac{\partial}{\partial\theta} F_i(\theta; \alpha_i, \beta_i) \qquad \nabla^2 \log\{\pi(\theta)\} = \frac{\partial^2}{\partial\theta^2} F_i(\theta; \alpha_i, \beta_i),$$

to determine α_i and β_i. For example, in the case of $\theta \in \mathbb{R}^k$, we use either the multivariate Gaussian or the multivariate Cauchy. For the former, the update equation leads to the natural proposal,

$$\mathcal{N}(\mu = \theta - \nabla^2 \log\{\pi(\theta)\}^{-1} \nabla \log\{\pi(\theta)\}, \Sigma = -\nabla^2 \log\{\pi(\theta)\}^{-1}).$$

The update term in the mean of this multivariate Gaussian is precisely the update term of the Newton-Raphson Method (Whittaker and Robinson, 1967), which is where NMC gets its name from. In case the estimated Σ has a negative eigenvalue we set those negative eigenvalues to a very small positive number, and reconstruct Σ.

The full list of estimation methods are enumerated in Appendix A. For example, for positive real values we use a Gamma proposer,

$$\text{Gamma}(\alpha = 1 - x^2 \nabla^2 \log\{\pi(x)\}, \beta = -x\nabla^2 \log\{\pi(x)\} - \nabla \log\{\pi(x)\}),$$

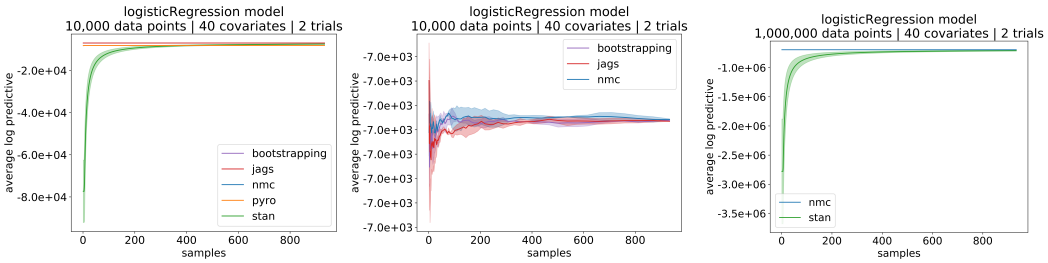
and we don't need a log-transform to an unconstrained space. In case multiple distributions can be fit, we pick the one which assigns the highest log-probability to the current state. Even though we may pick a different proposer at each point in the state space, the choice of this proposer is a deterministic function of θ, and so we can precisely compute the MH acceptance probability. We rely on generic Tensor libraries such as PyTorch (Paszke et al., 2017) that make it easy to write statistical models and also automatically compute the gradients. This makes our approach easy to apply to models generically.

An important observation related to our method is that we don't need to compute the Hessian of all the parameters in the latent space. Most statistical models can be decomposed into multiple latent variables. This decomposition allows for single site MCMC methods

that change the value of one variable at a time. In this case, we only need to compute the gradient and Hessian of the target density w.r.t. the variable being modified. Consider a model with N variables each drawn from R^K. The full Hessian is of size $(NK)^2$ and has a cost of $(NK)^3$ to invert. On the other hand, a single site approach computes N Hessians each of size K^2 with a total cost of NK^3 to invert.

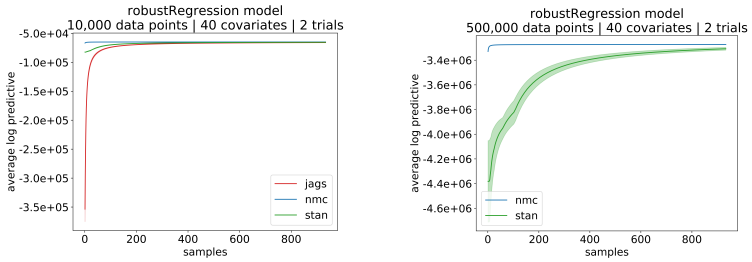
In the case of conjugate models, our estimation methods automatically recover the appropriate conditional posterior distribution, such as the ones used in BUGS (Spiegelhalter et al., 1996). However, even in cases of non-conjugacy, our proposal distributions pick out reasonable approximations to the conditional posterior of each variable.

3. Results



(a) 10,000 data points (b) Zoomed in view (c) 1,000,000 data points

Figure 1: Bayesian Logistic Regression model.

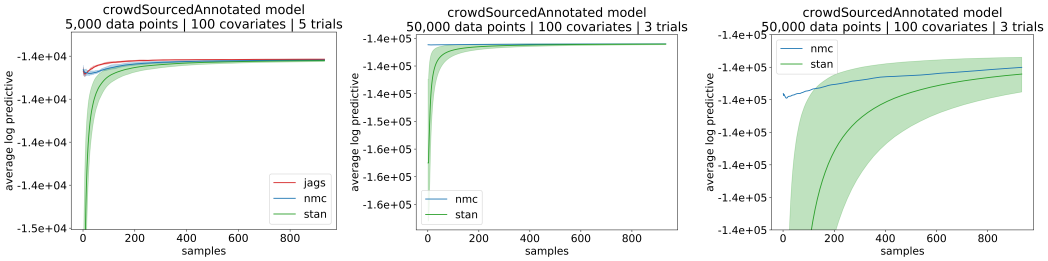


(a) 10,000 data points (b) 500,000 data points

Figure 2: Robust Regression model.

We present results of our experiments on three models – Bayesian Logistic Regression (Appendix B.1), Robust Regression (Appendix B.2), and a Crowd-Sourced Annotation Model (Passonneau and Carpenter (2014), Dawid and Skene (1979), Appendix B.3). In each experiment, we drew a sample of the latent variables from the model and N observed variables. Half of the observed variables were given to each inference engine, and the other half were used to compute the predictive likelihood over the posterior samples.

Figure 1 shows the relative convergence speed of various PPLs including Pyro (Bingham et al., 2019) and Stan on Bayesian Logistic Regression, which has a relatively easy log-



(*a*) 5000 data points (*b*) 50,000 data points (*c*) Zoomed in view

Figure 3: Crowd Sourced Annotation model.

Method	**N**	**Time**
NMC	20K	18
Stan	20K	41
JAGS	20K	2440
Bootstrap	20K	50
Pyro	20K	3024
NMC	2M	1030
Stan	2M	4900

(*a*) Logistic Regression

Method	**N**	**Time**
NMC	20K	68
Stan	20K	39
JAGS	20K	967
NMC	1M	1777
Stan	1M	3500

(*b*) Robust Regression

Method	**N**	**Time**
NMC	10K	71
Stan	10K	247
JAGS	10K	28
NMC	100K	430
Stan	100K	2900

(*c*) Annotation Model

Table 1: Wall clock time in seconds to generate 1000 samples.

concave posterior. All of the inference engines except for Stan converge to the true posterior fairly quickly in terms of samples. However, most of them are very slow (Table 1) and only Stan and NMC could be run on a larger data set. NMC is nearly 5 times faster than Stan, which uses NUTS, in addition to converging faster.

Robust Regression, on the other hand, doesn't have a log-concave posterior and both JAGS (Plummer et al., 2003) and Stan struggle to converge (Figure 2). In fact, Stan takes twice as much time for the same number of samples and it doesn't appear to have converged.

The final model, Crowd-Sourced Annotation Model, is a classic hierarchical statistical model. Each random variable has a conjugate conditional posterior, and since JAGS is designed to exploit conjugacy it really shines in this example. Unfortunately, the version of JAGS that we used kept crashing on larger data sets. Figure 3 shows that NMC is easily able to keep up with JAGS in terms of number of samples and is only a factor of 2.5 slower on the small data set. On the larger data set, NMC is nearly 7 times faster than Stan.

4. Conclusion

We have presented a novel MCMC method that uses the curvature of the target density to converge faster than existing state of the art methods, and without requiring any adaptive tuning. As next steps, we will fully integrate NMC into a production PPL and evaluate its performance across a wider spectrum of illustrative and real-world use cases.

References

Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Thomas P Minka. Beyond Newton's method, 2000.

Radford M Neal. Bayesian learning via stochastic dynamics. In *Advances in neural information processing systems*, pages 475–482, 1993.

Jorge Nocedal and Stephen J Wright. Numerical optimization second edition. *Numerical optimization*, pages 497–528, 2006.

Rebecca J Passonneau and Bob Carpenter. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326, 2014.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Martyn Plummer et al. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd international workshop on distributed statistical computing*, volume 124, page 10. Vienna, Austria., 2003.

G Robert and R Tweedie. Exponential convergence of Langevin diffusions and their discrete approximation. *Bernoulli*, 2:341–363, 1996.

Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

Gareth O Roberts and Osnat Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.

Umut Simsekli, Roland Badeau, Taylan Cemgil, and Gaël Richard. Stochastic quasi-newton langevin monte carlo. In *International Conference on Machine Learning (ICML)*, 2016.

David Spiegelhalter, Andrew Thomas, Nicky Best, and Wally Gilks. BUGS 0.5: Bayesian inference using Gibbs sampling manual (version ii). *MRC Biostatistics Unit, Institute of Public Health, Cambridge, UK*, pages 1–59, 1996.

E T Whittaker and George Robinson. The Newton-Rhapson method. *The Calculus of Observations; a Treatise on Numerical Mathematics, 4th ed*, page 84–87, 1967.

Yichuan Zhang and Charles A Sutton. Quasi-newton methods for Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 2393–2401, 2011.

Appendix A. Estimation of Probability Distributions

The estimation rules presented here are based on the work of Minka (2000).

A.1. Unconstrained spaces

In this section, we will consider distributions with the support $\mathbb{R}^k$.

A.1.1. Normal Distribution

The multivariate Normal distribution has the log-density:

$$\text{Normal}(x;\mu,\Sigma) = \text{const}(\mu,\Sigma) - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu).$$

Thus,

$$\begin{aligned}\frac{\partial}{\partial x}\text{Normal}(x;\mu,\Sigma) &= -\Sigma^{-1}(x-\mu), \text{and}\\ \frac{\partial^2}{\partial x^2}\text{Normal}(x;\mu,\Sigma) &= -\Sigma^{-1}.\end{aligned}$$

This leads to the natural estimation rule:

$$\begin{aligned}\mu &= x - \nabla^2 \log\{\pi(x)\}^{-1}\nabla \log\{\pi(x)\},\\ \Sigma &= -\nabla^2 \log\{\pi(x)\}^{-1}.\end{aligned}$$

A.1.2. Cauchy Distribution

The multivariate Cauchy distribution has the log-density:

$$\text{Cauchy}(x; b, A) = \text{const}(b, A) - \log(1 + (x - b)^T A(x - b))$$

Thus,

$$\frac{\partial}{\partial x}\text{Cauchy}(x; b, A) = \frac{-2A(x - b)}{1 + (x - b)^T A(x - b)}, \text{and}$$
$$\frac{\partial^2}{\partial x^2}\text{Cauchy}(x; b, A) = \frac{-2A}{1 + (x - b)^T A(x - b)} + \frac{4A(x - b)(x - b)^T A}{(1 + (x - b)^T A(x - b))^2}.$$

Noting that the second term above is the outer product of the first gradient leads to the following estimation rules:

$$b = x - \left(\nabla^2 \log\{\pi(x)\} - \nabla \log\{\pi(x)\} \nabla \log\{\pi(x)\}^T\right)^{-1} \nabla \log\{\pi(x)\},$$
$$s = \nabla \log\{\pi(x)\}^T \left(\nabla^2 \log\{\pi(x)\}\right)^{-1} \nabla \log\{\pi(x)\},$$
$$A = \left(\nabla^2 \log\{\pi(x)\} - \nabla \log\{\pi(x)\} \nabla \log\{\pi(x)\}^T\right) \frac{s - 1}{2 - s}.$$

A.2. Half Spaces

Half spaces refer to $\mathbb{R}^+$. For example, the Gamma distribution, which has the log-density:

$$Gamma(x; \alpha, \beta) = \text{const}(\alpha, \beta) + (\alpha - 1) \log x - \beta x.$$

Thus,

$$\frac{\partial}{\partial x} Gamma(x; \alpha, \beta) = \frac{\alpha - 1}{x} - \beta,$$
$$\frac{\partial^2}{\partial x^2} Gamma(x; \alpha, \beta) = -\frac{\alpha - 1}{x^2}.$$

Which leads to the estimation rules:

$$\alpha = 1 - x^2 \nabla^2 \log\{\pi(x)\},$$
$$\beta = -x \nabla^2 \log\{\pi(x)\} - \nabla \log\{\pi(x)\},$$

A.3. Simplexes

The K-simplexes refers to the set $\{x \in \mathbb{R}^{+K} | \sum_{i=1}^K x_i = 1\}$. We use the Dirichlet distribution to propose random variables with this support. The log-density of the Dirichlet is given by,

$$\text{Dir}(x; \alpha) = \text{const}(\alpha) + \sum_{i=1}^K (\alpha_i - 1) \log(x_i).$$

We consider the modified density, which includes the simplex constraint,

$$\mathrm{Dir}(x;\alpha) = \mathrm{const}(\alpha) + \sum_{i=1}^{K}(\alpha_i - 1)\log\frac{x_i}{\sum_{j=1}^{K} x_j}.$$

Thus,

$$\frac{\partial}{\partial x_i}\mathrm{Dir}(x;\alpha) = \frac{(\alpha_i - 1)}{x_i} - \frac{\sum_{j=1}^{K}(\alpha_j - 1)}{\sum_{j=1}^{K} x_j}, \text{and}$$
$$\frac{\partial^2}{\partial x_i \partial x_l}\mathrm{Dir}(x;\alpha) = -\delta_{il}\frac{(\alpha_i - 1)}{x_i^2} + \frac{\sum_{j=1}^{K}(\alpha_j - 1)}{(\sum_{j=1}^{K} x_j)^2}.$$

Which leads to the following robust estimation rule,

$$\alpha_i = 1 - x_i^2\left(\nabla_{ii}^2 \log\pi(x) - \max_{j\neq i}\nabla_{ij}^2 \log\pi(x)\right).$$

Appendix B. Statistical Models Used in Experiments

In all the models, N is typically the number of observed variables and K is roughly the number of latent variables, referred to as covariates in Section 3.

B.1. Bayesian Logistic Regression

$$\begin{aligned}
\alpha &\sim \mathcal{N}(0, 10, \mathrm{size} = 1),\\
\beta &\sim \mathcal{N}(0, 2.5, \mathrm{size} = K),\\
X_i &\sim \mathcal{N}(0, 10, \mathrm{size} = K) \quad \forall i \in 1..N\\
\mu_i &= \alpha + {X_i}^T\beta \quad \forall i \in 1..N\\
Y_i &\sim \mathrm{Bernoulli}(\mathrm{logit} = \mu_i) \quad \forall i \in 1..N.
\end{aligned}$$

X_i and Y_i are observed.

B.2. Robust Regression

$$\begin{aligned}
\nu &\sim \mathrm{Gamma}(2,\ 0.1)\\
\sigma &\sim \mathrm{Exponential}(\sigma_{mean})\\
\alpha &\sim \mathrm{Normal}(0,\ \sigma = \alpha_{scale})\\
\beta &\sim \mathrm{Normal}(\beta_{loc},\ \sigma = \beta_{scale},\ \mathrm{size} = K)\\
X_i &\sim \mathrm{Normal}(0,\ 10,\ \mathrm{size} = K) \quad \forall i \in 1\ldots N\\
\mu_i &= \alpha + \beta^T x \quad \forall i \in 1\ldots N\\
Y_i &\sim \text{Student-T}(\nu,\ \mu_i,\ \sigma) \quad \forall i \in 1\ldots N
\end{aligned}$$

X_i and Y_i are observed.

B.3. Annotation Model

There are N items, K labelers, and each item could be one of C categories. Each item i is labeled by a set J_i of labelers. z_i is the true label for item i and y_{ij} is the label provided to item i by labeler j. Each labeler l has a confusion matrix θ_l such that θ_{lmn} is the probability that an item with true class m is labeled n by l.

$$\begin{aligned}
\pi &\sim \text{Dirichlet}(\frac{1}{C}, \ldots, \frac{1}{C}) \\
z_i &\sim \text{Categorical}(\pi) \quad \forall i \in 1 \ldots N \\
\theta_{lm} &\sim \text{Dirichlet}(\alpha_m) \quad \forall l \in 1 \ldots K,\ m \in 1 \ldots C \\
|J_i| &\sim \text{Poisson}(J_{loc}) \\
l \in J_i &\sim \text{Uniform}(1 \ldots K) \\
y_{il} &\sim \text{Categorical}(\theta_{lz_i}) \quad \forall l \in J_i
\end{aligned}$$

Here $\alpha_m \in \mathbb{R}^{+C}$. We set $\alpha_{mn} = \gamma \cdot \rho$ if $m = n$ and $\alpha_{mn} = \gamma \cdot (1 - \rho) \cdot \frac{1}{C-1}$ if $m \neq n$. Where γ is the concentration and ρ is the *a-priori* correctness of the labelers. In this model, y_{il} and J_i are observed. In our experiments we set $\gamma = 10$ and $\rho = 0.5$.