Stochastic gradient MCMC algorithms and their applications

Ayoub El hanchi

Supervisor: Professor David A. Stephens

September 13, 2018

1 Introduction and Motivation

Markov chain Monte Carlo (MCMC) methods are a class of algorithms that attempt to draw independent samples from an arbitrary distribution given its (unnormalized) probability density function. These methods work by constructing a Markov chain that has the desired distribution as its stationary distribution. Classical MCMC methods generally make no assumptions about the probability density function, and are thus very general. On the other hand, these methods usually require tuning the parameters of the Markov chain, namely the starting point of the chain and its transition kernel, to achieve good performance. This is in general difficult and has to be done manually for almost every new problem. A poor choice of the parameters can lead to:

1. Slow convergence: This can be caused by (a) Choosing a starting point that is very far from regions of high density. (b) Choosing a transition kernel that makes successive samples either too close or too far from each other. (”far” and ”close” here are relative to the variance of the distribution to be sampled from)

2. Correlation between successive samples: Mainly caused by (b) above. Very close samples are likely to be correlated, while very far samples are likely to be rejected.

Gradient-based MCMC methods attempt to use the local structure of the density function to overcome these problems by assuming that the local behavior of the density can be extrapolated to a larger useful scale. Beyond the fact that these methods are slightly less general than the classical MCMC methods, they also require additional computing effort. In various applications, the computation of the gradient can be prohibitively expensive. In these cases, using a stochastic gradient can be the only viable option.

In this research project, we explore stochastic gradient MCMC methods, and their applications in selected statistical problems such as Bayesian inference and maximum likelihood estimation in mixed models. We start by carefully constructing these algorithms, making sure we keep track of our assumptions. We then apply the developed algorithms to real datasets and present our results.

2 Theory

While most classical MCMC algorithms rely on discrete-time Markov processes, gradient-based methods rely on their continuous-time analog. On the one hand, it is extremely difficult to model gradient-based methods as discrete-time Markov processes. On the other, moving to continuous-time allows gradient-based methods, at least in theory, to continuously adapt to the geometry of the density function as the process progresses, explicitly taking into account the fact that the gradient is a local property of the density function.

First, given a probability density function of some target distribution, we will show how to construct a continuous Markov process that has the target distribution as its stationary distribution. Second, we will show how a stochastic gradient can replace the full gradient in the process without altering its stationary distribution. Finally, we will review our assumptions and focus on the important special case of approximately quadratic density functions, for which we develop a new algorithm.

2.1 Construction

2.1.1 Continuous Markov Processes that converge to target distribution

Let $\boldsymbol{\theta}$ be a random variable distributed according to the probability density function $f(\boldsymbol{\theta})$. (Yi-An Ma et al, 2015) [2] showed that *all* continuous Markov processes with random variables $\boldsymbol{z_t} := (\boldsymbol{\theta_t}, \boldsymbol{r_t})$ and stationary distribution $h(\boldsymbol{\theta}, \boldsymbol{r})$ that have $f(\boldsymbol{\theta})$ as the marginal stationary distribution of $\boldsymbol{\theta}$ are of the form:

$$d\boldsymbol{z} = \boldsymbol{g(z)}dt + \sqrt{2\boldsymbol{D(z)}}d\boldsymbol{W} \tag{1}$$

where:

$$\boldsymbol{g(z)} = [\boldsymbol{D}(\boldsymbol{z}) + \boldsymbol{Q}(\boldsymbol{z})]\nabla \log(h(\boldsymbol{z})) + \boldsymbol{\Gamma}(\boldsymbol{z})$$

$$\boldsymbol{\Gamma}_i(z) = \sum_{i=1}^{d} \frac{\partial}{\partial z_j}[\boldsymbol{D_{ij}}(\boldsymbol{z}) + \boldsymbol{Q_{ij}}(\boldsymbol{z})]$$

with $\boldsymbol{D(z)}$ positive (semi)definite, and $\boldsymbol{Q(z)}$ skew-symmetric.

Therefore, the family of all continuous Markov processes converging to the desired distribution can be parametrized by matrices $\boldsymbol{D(z)}$, $\boldsymbol{Q(z)}$ and a choice of $\boldsymbol{z}$. We will refer to these parameters as the free parameters of the process. The authors note that the known gradient-based methods only explore a small portion of this family of processes. One interesting topic of future research is to consider the problem of finding a set of "optimal" free parameters, where optimality can be determined by the convergence rate of the process, the sensibility of its convergence to various errors, or other useful criteria.

2.1.2 Discretization of the process

To be able to simulate a continuous Markov process, we resort to using ϵ-discretization. Making the substitution $dt \to \epsilon$ for some $0 < \epsilon \ll 1$, the discretized process can be written as:

$$\Delta\boldsymbol{\theta} = \epsilon([\boldsymbol{D}(\boldsymbol{\theta}) + \boldsymbol{Q}(\boldsymbol{\theta})]\nabla_{\boldsymbol{\theta}} \log(f(\boldsymbol{\theta})) + \boldsymbol{\Gamma}(\boldsymbol{\theta})) + \mathcal{N}(0, 2\epsilon\boldsymbol{D}(\boldsymbol{\theta})) \tag{2}$$

where $\boldsymbol{\Gamma}(\theta)$ is defined as in (1).

Discretization alters the stationary distribution of the process. Its effect decreases the smaller we set ϵ to be. On the other hand, examining equation (2), we notice that the total noise is directly proportional to ϵ, hence shrinking ϵ to zero makes successive samples extremely correlated when the size of the gradient is small.

In general, it is difficult to rigorously quantify this error. All we can say is that it is inversely proportional to the sizes of $\boldsymbol{D}(\boldsymbol{\theta})$, $\boldsymbol{Q}(\boldsymbol{\theta})$, $\boldsymbol{\Gamma}(\boldsymbol{\theta})$, and $\nabla_{\boldsymbol{\theta}} \log(f(\boldsymbol{\theta}))$. For the rest of this report, we will assume $\epsilon := 1$ for simplicity, unless stated otherwise. One way we can attempt to quantify this error is by modeling (2) as a discrete Markov process. We can then try to bound its deviation from the target distribution. This is indeed the approach we take to quantify this error in the special case of approximately quadratic log densities and constant $\boldsymbol{D}(\boldsymbol{\theta}) = \boldsymbol{D}$ and $\boldsymbol{Q}(\boldsymbol{\theta}) = \boldsymbol{Q}$. Unfortunately, as mentioned before, this approach is in general very difficult, and in most cases we cannot solve for the stationary distribution of the discrete Markov process.

2.1.3 Replacing the full gradient with a stochastic gradient

In the context of Bayesian inference, (Yi-An Ma et al, 2015) [2] also showed how to replace the gradient of the log density with a stochastic gradient by taking into account the noise added by the use of a stochastic gradient. We reproduce their method here for completeness, while being a little more explicit about our assumptions, and dealing with some issues not addressed in the original paper. For the remaining of this section we will assume $\boldsymbol{z} := \boldsymbol{\theta}$.

Suppose we are given data x with $N \gg 1$ observations, and an unnormalized posterior density $p(\boldsymbol{\theta}|x) = \prod_{i=1}^{N} p(x_i|\boldsymbol{\theta})p(\boldsymbol{\theta})$, and we would like to sample from this posterior. Following (1), we need to compute $\nabla \log p(\boldsymbol{\theta}|x) = \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}}(\log p(x_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}))$. Define the stochastic gradient to be:

$$\nabla^S_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|X) = N\frac{1}{|S|}\sum_{j\in S} \nabla_{\boldsymbol{\theta}}(\log p(x_j|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})) \tag{3}$$

where $S \subset \{1, ..., N\}$ is random. The size of S is usually chosen to be small such that the computation of the stochastic gradient is inexpensive.

Let $U(\boldsymbol{\theta}) := \nabla_{\boldsymbol{\theta}}(\log p(X|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}))$, where X is a random variable distributed according to the empirical distribution generated by $\{x_1, ..., x_N\}$. Then we have:

$$\frac{1}{N}\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x) = E_X[U(\boldsymbol{\theta})]$$

Define $C(\boldsymbol{\theta}) = \mathrm{Var}_X[U(\boldsymbol{\theta})]$. Assuming $|S|$ is large enough for the central limit theorem to be applicable:

$$\frac{1}{N}\nabla^S_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|X) \approx E_X[U(\boldsymbol{\theta})] + \mathcal{N}(0, \frac{1}{|S|}C(\boldsymbol{\theta})) = \frac{1}{N}\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x) + \mathcal{N}(0, \frac{1}{|S|}C(\boldsymbol{\theta}))$$

and therefore:

$$\nabla^S_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x) - \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x) \sim \mathcal{N}(0, \frac{N^2}{|S|}C(\boldsymbol{\theta})) \tag{4}$$

Replacing the gradient with the stochastic gradient in equation (2), and removing the added noise, we arrive at the following update equation:

$$\Delta\boldsymbol{\theta} = ([\boldsymbol{D}(\boldsymbol{\theta}) + \boldsymbol{Q}(\boldsymbol{\theta})]\nabla^S_{\boldsymbol{\theta}} \log(f(\boldsymbol{\theta})) + \boldsymbol{\Gamma}(\theta)) + \mathcal{N}(0, 2\boldsymbol{D}(\boldsymbol{\theta}) - \boldsymbol{B}(\boldsymbol{\theta})) \tag{5}$$

where

$$\boldsymbol{B}(\boldsymbol{\theta}) = \frac{N^2}{|S|}[\boldsymbol{D}(\boldsymbol{\theta}) + \boldsymbol{Q}(\boldsymbol{\theta})]C(\boldsymbol{\theta})[\boldsymbol{D}(\boldsymbol{\theta}) + \boldsymbol{Q}(\boldsymbol{\theta})]$$

for some arbitrary choice of $\boldsymbol{D}(\boldsymbol{\theta})$ and $\boldsymbol{Q}(\boldsymbol{\theta})$, and where it is assumed that $2\boldsymbol{D}(\boldsymbol{\theta}) \succeq \boldsymbol{B}(\boldsymbol{\theta})$, and that an estimate of $\boldsymbol{C}(\boldsymbol{\theta})$ is available.

2.2 Assumptions and Limitations

Our assumptions of the last section are:

1. $|S|$ is large enough for the central limit theorem approximation to hold.
2. $\boldsymbol{C}(\boldsymbol{\theta})$ can be accurately estimated.
3. $2\boldsymbol{D}(\boldsymbol{\theta}) \succeq \boldsymbol{B}(\boldsymbol{\theta})$.
4. The ϵ-discretization error is negligible.

The limitations of this method are therefore dictated by how well we can satisfy these assumptions. In practice, assumption 1 holds almost always for $|S| \geq 50$ and can therefore be safely assumed to be true. Assumptions 2 and 3 are on the other hand difficult to satisfy in part because of their dependence on $\boldsymbol{\theta}$.

To see why the estimation of $\boldsymbol{C}(\boldsymbol{\theta})$ is difficult, one has to remember that we are only allowed to use a small subset S satisfying $|S| \ll N$ of the data x. The sample variance computed using that small subset S is usually not accurate enough to preserve the dynamics of the Markov process, and can be a significant cause of error.

In some rare cases, it is possible to derive an analytical expression for $\boldsymbol{C}(\boldsymbol{\theta})$ in terms of $\mathrm{Var}[X]$, which can be computed exactly, and which can be done only once for the whole process, preserving the efficiency gained by using the stochastic gradient. No solution to this problem is known for the general case to my knowledge.

Once assumption 2 is satisfied, one can then set $\boldsymbol{D}(\boldsymbol{\theta})$ to satisfy assumption 3. In addition to the computational burden of determining $\boldsymbol{\Gamma}(\boldsymbol{\theta})$ (and the possible analytical intractability of this problem), the challenge in this case is to control the injected noise such that the total noise of the process is of the same order as the variance of the target distribution. Achieving this decreases the correlation between successive samples and increases the rate of convergence of the process. This is in general very challenging since the variance of the target distribution is usually not available (and estimating it might even be the goal of using the algorithm in the first place). Nonetheless, in some cases, one can make use of the local structure of the density function to have an estimate of the size of this variance. (One can think of using a Taylor approximation in a region near the mode for example, assuming the distribution is unimodal).

For a discussion of assumption 4, see section 2.1.2..

2.3 Special case: Approximately Quadratic log densities

We consider the case of an approximately quadratic log density:

$$-\log f(\boldsymbol{\theta}) \approx -\log f(\boldsymbol{\theta}_{MAP}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_{MAP})^T H(\boldsymbol{\theta} - \boldsymbol{\theta}_{MAP}) \tag{6}$$

where $\boldsymbol{\theta}_{MAP}$ is the minimum of the negative log density of the posterior, and $\boldsymbol{H^{-1}}$ is the inverse Hessian at $\boldsymbol{\theta}_{MAP}$. Invoking the Bernstein-Von Mises theorem, this quadratic approximation holds in our setup ($N \gg 1$), provided that the contribution of the likelihood to the density dominates that of the prior distribution. More generally, if the prior distribution is also well approximated by a Gaussian, the approximation (6) holds regardless of the strength of the prior. We consider here this more general setup. Our goal will then be to develop a stochastic gradient MCMC algorithm for this important special case.

We start by showing how to satisfy the assumptions of the previous section. We then develop a simple criterion to assess the convergence of the process. We end our discussion by presenting a new algorithm that brings all of our ideas together.

2.3.1 Estimating $\boldsymbol{C}(\boldsymbol{\theta})$ and choosing $\boldsymbol{D}(\boldsymbol{\theta})$ and $\boldsymbol{Q}(\boldsymbol{\theta})$

To estimate $\boldsymbol{C}(\boldsymbol{\theta})$, notice that since N is large, the Hessian of the likelihood is approximately constant, and we have:

$$-\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x) = E_X[-\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|X)] = \mathrm{Var}_X[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|X)] = N\boldsymbol{C}(\boldsymbol{\theta}) = N\boldsymbol{C}$$

and therefore $\boldsymbol{C}(\boldsymbol{\theta}) = \boldsymbol{C}$ is approximately constant, and can be efficiently estimated at the start of the algorithm. (if data is provided in an online manner, one can compute a running average instead). This satisfies assumption 2 in the previous section.

Following our discussion of assumption 3 in the previous section, it remains to choose $\boldsymbol{D}(\boldsymbol{\theta})$ and $\boldsymbol{Q}(\boldsymbol{\theta})$ such that the computation of $\boldsymbol{\Gamma}(\boldsymbol{\theta})$ is inexpensive, and such that the total noise is positive semi-definite and of the same order as the variance of the target distribution. In this case, both requirements can be satisfied.

First, the fact that $\boldsymbol{C}(\boldsymbol{\theta}) = \boldsymbol{C}$ is constant allows us to set $\boldsymbol{D}(\boldsymbol{\theta}) = \boldsymbol{D}$ and $\boldsymbol{Q}(\boldsymbol{\theta}) = \boldsymbol{Q}$ to be constants and still be able to satisfy assumption 3, which makes $\boldsymbol{\Gamma}(\boldsymbol{\theta}) = 0$. We also set for simplicity $\boldsymbol{Q} = \boldsymbol{0}$. We therefore consider updates of the form:

$$\Delta\boldsymbol{\theta} = \boldsymbol{D}\ \nabla^S_{\boldsymbol{\theta}} \log(f(\boldsymbol{\theta})) + \mathcal{N}(0, 2\boldsymbol{D} - \frac{N^2}{S}\boldsymbol{DCD}) \tag{7}$$

Before dealing with the issue of positive definiteness of the noise, we first consider the discretization error. In the Appendix, we quantify this error, and we show that the above update equation deviates from the posterior. We also show that the correct update equation is given by:

$$\Delta\boldsymbol{\theta} = \boldsymbol{D}\ \nabla^S_{\boldsymbol{\theta}} \log(f(\boldsymbol{\theta})) + \mathcal{N}(0, 2\boldsymbol{D} - \frac{N^2}{S}\boldsymbol{DCD} - \boldsymbol{DHD}) \tag{8}$$

where $\boldsymbol{H}$ is the Hessian at the minimum, and which can be approximated by (9).

It remains to choose $\boldsymbol{D}$ such that the total noise is positive definite and of the same order as the variance. By the quadratic assumption, $\mathrm{Var}_{\boldsymbol{\theta}|x}[\boldsymbol{\theta}] \approx \boldsymbol{H^{-1}}$ and letting $\boldsymbol{G}(\boldsymbol{\theta}) := -\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta})$ we have:

$$\boldsymbol{H} = -\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x) - \nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) \approx N\boldsymbol{C} + \boldsymbol{G}(\boldsymbol{\theta}) = N\boldsymbol{C} + \boldsymbol{G} \tag{9}$$

Setting the total noise to be a fraction of the variance of the target distribution:

$$\boldsymbol{D} = \frac{k}{2}\boldsymbol{H}^{-1} = \frac{k}{2}(N\boldsymbol{C} + \boldsymbol{G})^{-1} \text{ for some } k \in (0, 1) \tag{10}$$

The restriction on the values of k is necessary for convergence, as we show in the Appendix. To enforce assumption 3, we need:

$$\boldsymbol{H}^{-1} - \frac{k}{4}\frac{N}{|S|}\boldsymbol{H}^{-1}N\boldsymbol{C}\boldsymbol{H}^{-1} - \frac{k}{4}\boldsymbol{H}^{-1} \succeq 0 \tag{11}$$

We have:

$$\begin{aligned} &\boldsymbol{H} \succeq N\boldsymbol{C} \\ \Leftrightarrow\ &\boldsymbol{H}^{-1}\boldsymbol{H}\boldsymbol{H}^{-1} \succeq \boldsymbol{H}^{-1}N\boldsymbol{C}\boldsymbol{H}^{-1} \\ \Leftrightarrow\ &\boldsymbol{H}^{-1} \succeq \boldsymbol{H}^{-1}N\boldsymbol{C}\boldsymbol{H}^{-1} \end{aligned} \tag{12}$$

So that (11) holds if:

$$(1 - \frac{k}{4}\frac{N + |S|}{|S|})\boldsymbol{H}^{-1} \succeq 0$$

hence :

$$k \leq 4\frac{|S|}{N + |S|} \Rightarrow 2\boldsymbol{D} - \frac{N^2}{S}\boldsymbol{D}\boldsymbol{C}\boldsymbol{D} - \boldsymbol{D}\boldsymbol{H}\boldsymbol{D} \succeq 0 \tag{13}$$

Therefore, we can guarantee assumption 3 by letting $k \leq 4\frac{|S|}{N+|S|}$.

To reduce autocorrelation and increase the rate of convergence of the process, a large total noise is desirable (see Appendix for rigorous justification). In cases where $|S| \ll N$, the variance of the total noise is only a small fraction of the variance of the target distribution, which slows down the convergence of the process. Our next task will be to derive a higher upper bound on k. We split our analysis in two cases.

2.3.1.1 Weak prior

In the case $\boldsymbol{G} \ll N\boldsymbol{C}$, we have $\boldsymbol{H} \approx N\boldsymbol{C}$ which implies that the bound (12) is saturated, and that the condition in (13) becomes a necessary one. Concretely, this means that the total noise in this setup cannot be greater than $\approx 2\frac{|S|}{N}\mathrm{Var}_{\boldsymbol{\theta}|x}[\boldsymbol{\theta}]$. To maximize the rate of convergence, we set $k = 4\frac{|S|}{N+|S|}$ and we get the update equation:

$$\Delta\boldsymbol{\theta} = 2\frac{|S|}{(N + |S|)N}\boldsymbol{C^{-1}}\nabla^S_{\boldsymbol{\theta}}\log(f(\boldsymbol{\theta})) \tag{14}$$

which is, in the limit $\frac{|S|}{N} \to 0$, the same update equation proposed by (Mandt et al., 2017) [3] for approximate inference with SGD using less general arguments. This is also a special case of stochastic gradient fisher scoring (SGFS) first proposed in [1]. Our derivation shows that this algorithm is "optimal" in terms of convergence and reduced auto-correlation in the family of SGFS algorithms, when applied to problems with a weak prior, since it uses the maximum allowable amount of total noise. We show in the Appendix, that in the case of a strong prior (which is often the case in hierarchical models), the process described by (14) might not even have a stationary distribution.

2.3.1.2 Strong prior

In the case of a strong prior, the bound (12) is weak, and we can derive a better upper bound on k. Another way to enforce positive semi-definiteness of (11) is by requiring:

$$(1 - \frac{k}{4})\lambda_{min}(\boldsymbol{H}^{-1}) \geq \frac{k}{4}\frac{N}{|S|}\lambda_{max}(\boldsymbol{H}^{-1}N\boldsymbol{C}\boldsymbol{H}^{-1})$$

$$\Leftrightarrow k \leq \frac{4r}{r + \frac{N}{|S|}} \tag{15}$$

with $r := \frac{\lambda_{min}(\boldsymbol{H}^{-1})}{\lambda_{max}(\boldsymbol{H}^{-1}N\boldsymbol{C}\boldsymbol{H}^{-1})}$ When $r = 1$, we recover the bound (13). For strong priors, $r > 1$, and we obtain a better upper bound. Using this new upper bound, we choose k as follows, for some constant $c \in (0, 1)$ of our choice (in practice, 0.1 works very well):

$$k = \min\{c, 4\max\{\frac{|S|}{N + |S|}, \frac{r}{r + \frac{N}{|S|}}\}\} \tag{16}$$

where $\boldsymbol{H}^{-1}$ is computed using (9) when computing r. Choosing k this way can dramatically increases the convergence of the process for very large datasets with strong priors. (There is still a gap between the two bounds I have derived, but I have not being able to come up with a better one without more assumptions. If the data set is very large, and the prior's variance $\boldsymbol{G}$ is of the same order as $N\boldsymbol{C}$, the convergence can be very slow if $|S| \ll N$. This is where I think a better bound can be useful.)

2.3.2 Convergence criterion

One of the issues usually ignored in the design of this type of algorithm is that of convergence. In this section we develop a simple criterion for detecting convergence, allowing us to start sampling only when we are guaranteed that the process has converged.

Consider the sequence $\{\boldsymbol{Y}_i\}_{i=1}^n$ of n successive stochastic gradients. By (4), we have:

$$\boldsymbol{Y}_i \sim \mathcal{N}(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x)|_{\theta=\theta_i}, \frac{N^2}{|S|}\boldsymbol{C}) \quad \forall i \in \{1, ..., n\}$$

Therefore:

$$\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{Y_i} \sim \mathcal{N}(\frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x)|_{\theta=\theta_i}, \frac{N^2}{n|S|}\boldsymbol{C})$$

Now:

$$\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x) \approx (\boldsymbol{\theta} - \boldsymbol{\theta}_{MAP})^T \boldsymbol{H} \approx (\boldsymbol{\theta} - \boldsymbol{\theta}_{MAP})^T (N\boldsymbol{C} + G)$$

Since $\boldsymbol{\theta}$ is approximately $\sim \mathcal{N}(\boldsymbol{\theta}_{MAP}, (N\boldsymbol{C} + \boldsymbol{G})^{-1})$, we have:

$$\frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|x)|_{\theta=\theta_i} \sim \mathcal{N}(0, \frac{1}{n}(N\boldsymbol{C} + \boldsymbol{G}))$$

and finally, we get:

$$\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{Y_i} \sim \mathcal{N}(0, \frac{1}{n}(N\boldsymbol{C} + \boldsymbol{G}) + \frac{N^2}{|S|n})$$

Hence, we can write the following convergence criterion, for some $b \in (0, 2]$:

$$\left|\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{Y_i}\right| \leq b\sqrt{\frac{1}{n}(N\boldsymbol{C} + \boldsymbol{G}) + \frac{N^2}{|S|n}} \tag{17}$$

While the condition (17) is neither a sufficient nor a necessary condition for convergence, for $b \leq 1$, it is extremely unlikely that the iterates of the process pass this criterion without being reasonably close to the mode of the density. That is because the quantity on the left in (17) will have a very large mean if the iterates are far from the mode. The smaller the b, the more confidence we have in stating this convergence, although from my experiments, setting $b = 1$ works perfectly well.

The only piece we have not discussed is how to choose n. This will depend on the value of k used. A reasonable choice would be $1/k$, so that the process is given enough iterates to explore the posterior. Note that the additional computational cost of checking this condition is negligible, and does not affect the efficiency of the algorithm.

2.3.3 Algorithm

We end our discussion by presenting our algorithm, which is an optimized version of SGFS. We call it, unimaginatively, "Optimized Stochastic Gradient Fisher Scoring" (OSGFS). We incorporate the convergence criterion (17), and choose $\boldsymbol{D}$ according to (10) and (16). First, define:

- g := function that returns a matrix where the i^{th} row is the gradient of the negative log likelihood given the ith data point.
- *cov* := function that returns the empirical covariance of the passed samples.
- i := iteration number.

Algorithm 1 Optimized Stochastic Gradient Fisher Scoring (OSGFS)

```
procedure OSGFS($\boldsymbol{\theta}_0$, $x$, $|S|$, $g$, m)    ▷ m is the number of samples requested
    $\boldsymbol{\theta} = \boldsymbol{\theta}_0$
    start sampling = FALSE
    while # samples $\leq m$ do
        indices = draw $|S|$ samples from (discrete) $\mathcal{U}(1, N)$
        gradients = $g(\boldsymbol{\theta}$, $x$[indices, ])
        $\boldsymbol{C} = (1 - \frac{1}{i})\boldsymbol{C} + (\frac{1}{i})$ cov(gradients)
        update $\boldsymbol{\theta}$ according to (7) using (10) and (16).
        if ($\neg$ start sampling $\wedge$ $i \geq n$ $\wedge$ (17) is satisfied) then start sampling = TRUE
        end if
        if (start sampling) then store current $\boldsymbol{\theta}$
        end if
    end while
    return collected samples
end procedure
```

Our derivation shows the power of this framework and its generality. It is interesting to study how the process changes when allowing $\boldsymbol{Q} \neq \boldsymbol{0}$, and if there exists in that case a more "optimal" process in terms of rate of convergence and reduced correlation, but we leave this question for future investigations.

3 Applications

We apply OSGFS on standard Bayesian inference, and compare its performance with that of standard MCMC methods. We then consider the problem of maximum likelihood estimation in mixed models. We start by reviewing the most commonly used methods and then discuss in which cases our algorithm can be useful.

3.1 Bayesian logistic regression

We consider the simple model:

- $y_i|\boldsymbol{\theta}, \boldsymbol{x_i} \sim Bernoulli(\text{logistic}(\boldsymbol{x_i}\boldsymbol{\theta}))$
- $\boldsymbol{\theta} \sim \mathcal{N}(0, 100\boldsymbol{I})$

We run our algorithm on 3 different data sets:

- The *Skin Segmentation data set* which has $N = 245\ 057$, 3 features, and a binary output variable. This is the same data set used my (Mandt, 2017).
- The *Cod RNA data set* which has $N = 488\ 566$, 8 features, and a binary output variable.
- The *Run or Walk data set* which has $N = 88\ 589$, 6 features, and a binary output variable.

For the experiments on the *Skin Segmentation* and *Cod RNA* data sets, we used $|S| = 10\ 000$ and collected 100 000 samples, while for the last experiment, we used $|S| = 1000$ and collected 10 000 samples. In all three cases, we collected 100 000 samples when running the Metropolis algorithm.

We also compare the output of the algorithm with and without the convergence criterion (17) for 10 000 iterations to illustrate its effect in Figure (2).

The usefulness of this method in simple models is limited, I think, to the online learning setting. When performing batch learning, a Gaussian approximation of the posterior is likely to be more efficient and easier to manipulate.

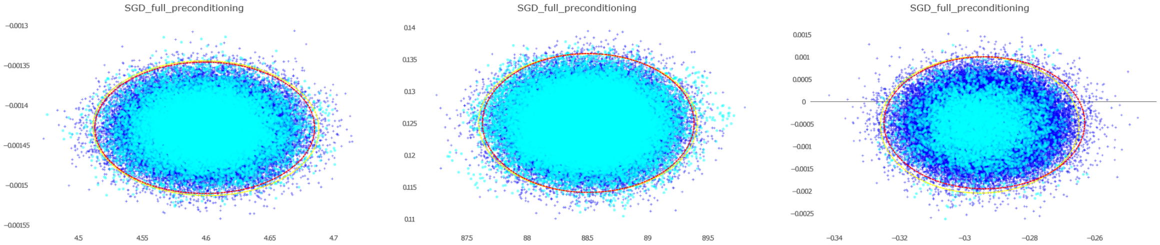

Figure 1: From left to right: *Skin Segmentation*, *Cod RNA*, *Run or Walk*. Samples from the Metropolis algorithm are shown in blue, with the red ellipse showing the covariance of the posterior (3 standard deviations). Samples from OSGFS are show in Cyan, and their covariance by the yellow ellipse. The samples displayed are projected onto the smallest and largest principal components of the posterior. We set $k = 4\frac{|S|}{N+|S|}$ since we have a very weak prior.

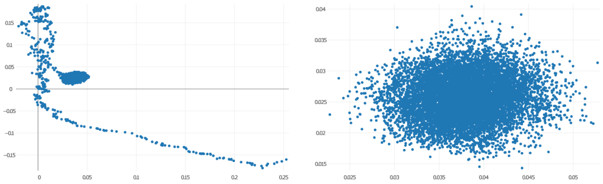

Figure 2: The same starting point is used in both runs. Conditioning the sampling by (17) (right picture) allows the algorithm to determine the right "burn-in" period, and to only collect samples from the posterior.

3.2 Bayesian logistic-normal Hierarchical model

We now consider the more complex model:

- $y_{ij}|\boldsymbol{\theta_i}, \boldsymbol{x_{ij}} \sim Bernoulli(\text{logistic}(\boldsymbol{x_{ij}\theta_i}))$
- $\boldsymbol{\theta_i} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- $\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu_0}, \frac{1}{\lambda}\boldsymbol{\Sigma})$
- $\boldsymbol{\Sigma} \sim \mathcal{W}^{-1}(\boldsymbol{\Phi}, \nu)$

Where the data x is split into K different groups, each of which is assumed to have a parameter $\boldsymbol{\theta_i}$. We split the *Skin Segmentation data set* into 25 separate groups, and we fit the above model by Gibbs sampling. The sampling of the $\boldsymbol{\theta_i}$'s is done using our algorithm, while the sampling of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is done directly using the conjugate prior property. We present our results graphically in Figure (3).

The alternative to using the OSGFS algorithm is to sample the $\boldsymbol{\theta_i}$'s from a Gaussian approximation of their marginal posterior. This requires the computation of the mode of the log marginal density, and the Hessian at that mode for each $\boldsymbol{\theta_i}$. The advantage of our method is that it performs both the optimization and the sampling simultaneously. On the other hand, multiple iterations of our algorithm are required to guarantee that the process has traversed the posterior entirely, which can hinder this apparent gain in efficiency. In the specific case of logistic regression, where the Hessian is easily computable and the size of each data group is not very large, our results suggest that the two methods have approximately the same efficiency. This may not hold in cases where the Hessian is difficult to compute or when the size of each data group is very large, in which case our method can be the more efficient one.

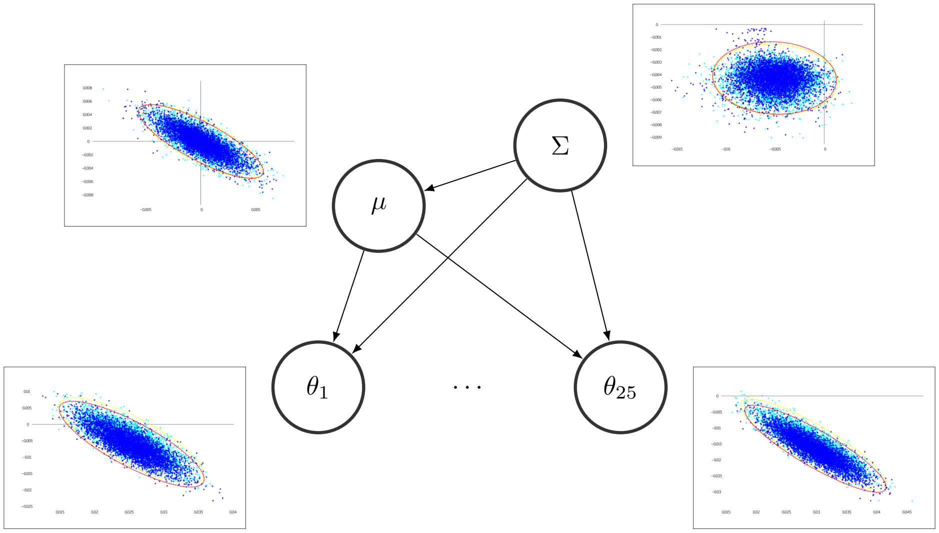

Figure 3: Logistic-normal hierarchical model fit with Gibbs sampling. Samples obtained using a Gaussian approximation of the marginal posteriors of the $\boldsymbol{\theta}_i$'s are shown in blue, with the red ellipses showing their covariance whereas samples obtained from OSGFS are show in Cyan, with the yellow ellipses showing their covariance. 5000 samples were obtained for each method. The samples shown for $\boldsymbol{\Sigma}$ are of two off-diagonal elements of its Cholesky decomposition, which are normally distributed.

3.3 Mixed models

3.3.1 Introduction

Mixed models are a class of statistical models used to analyze longitudinal and/or clustered data. They extend the class of fixed effects model, where the parameters of the model are assumed to be non-random, by including random effects, i.e, by assuming that some or all the parameters of the model have a fixed and random component. This allows mixed models to deal with data where the observations are identically distributed, but where the independence assumption between observations does not hold, as is the case for longitudinal and/or clustered data. An alternative to mixed models are hierarchical Bayesian models (like the one described in section 3.2), where, in the first level of the hierarchy, each cluster/subject is treated as a different set of observations with its own parameters, while at the second level, the parameters of each cluster/subject are assumed to be generated from a parent distribution. When the number of observations for each cluster/subject is small, the hierarchical Bayesian approach is not viable, but the mixed model approach is still applicable.

In the most general setting, a mixed model with N observations requires the specification of the distributions :

$$y_i \mid \boldsymbol{X_i}, \boldsymbol{\beta}, \boldsymbol{\mu}$$
$$\boldsymbol{\mu}$$

and assumes :

$$y_i \perp\!\!\!\perp y_j \mid \boldsymbol{X_i}, \boldsymbol{X_j}, \boldsymbol{\beta}, \boldsymbol{\mu} \quad \text{for all} \quad i \neq j$$

where:

y_i : i^{th} observation
$\boldsymbol{X_i}$: i^{th} $1 \times p$ vector of predictors
$\boldsymbol{\beta}$: $p \times 1$ vector of fixed effects
$\boldsymbol{\mu}$: $q \times 1$ vector of random effects with $E(\boldsymbol{\mu}) = \boldsymbol{0}$

Maximum likelihood is the main method used to estimate the parameters of mixed models. In the next sections, we review computational methods to find the maximum likelihood estimate for mixed models, and discuss how our algorithm can be used to perform this estimation. We focus our attention on two classes of mixed models: Linear mixed models (LMM's), and Generalized linear mixed models (GLMM's).

3.3.2 Linear mixed models

The linear mixed model assumes:

(a) $y_i \mid \boldsymbol{X_i}, \boldsymbol{\beta}, \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{X_i\beta} + \boldsymbol{Z_i\mu}, \sigma^2)$, where $\boldsymbol{Z_i}$ is a $1 \times q$ design vector that encodes the belonging of the i^{th} observation to its cluster(s)/subject.

(b) $\boldsymbol{\mu} \mid \boldsymbol{G}, \sigma \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{G})$

The model can be rewritten succinctly as:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{Z\mu} + \boldsymbol{\epsilon} \tag{18}$$

with $\boldsymbol{\epsilon} \mid \sigma \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I_N})$. Given that $\boldsymbol{\mu}$ and $\boldsymbol{\epsilon}$ are multivariate normal random variables, we immediately have from (18) :

$$\boldsymbol{y} \mid \boldsymbol{\beta}, \boldsymbol{G}, \sigma \sim \mathcal{N}(\boldsymbol{X\beta}, \sigma^2(\boldsymbol{ZGZ^T} + \boldsymbol{I_N})) \tag{19}$$

The parameters to estimate in this model are therefore $\boldsymbol{\beta}$, $\boldsymbol{G}$ and σ.

3.3.2.1 First order and quasi-Newton methods

In this section, we derive an analytical expression for the gradient of the negative profiled log likelihood function. We start with the negative log likelihood function (up to a constant). Let $\boldsymbol{D(G)} := \boldsymbol{ZGZ^T} + \boldsymbol{I_N}$. Then we have:

$$\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{G}, \sigma \mid \boldsymbol{y}) = -\log(P(\boldsymbol{y} \mid \boldsymbol{\beta}, \boldsymbol{G}, \sigma)) = N\log(\sigma) + \frac{1}{2}\log(\det(\boldsymbol{D})) + \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X\beta})^T\boldsymbol{D}^{-1}(\boldsymbol{y} - \boldsymbol{X\beta}) \tag{20}$$

We first minimize (20) with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\sigma}$ by setting the gradient to 0. We get:

$$\boldsymbol{\beta(D)} = \boldsymbol{(X^TD^{-1}X)^{-1}X^TD^{-1}y} \tag{21}$$

$$\boldsymbol{\sigma(D)} = \sqrt{\frac{1}{N}(\boldsymbol{y} - \boldsymbol{X\beta})^T\boldsymbol{D}^{-1}(\boldsymbol{y} - \boldsymbol{X\beta})} \tag{22}$$

Both $\boldsymbol{\beta}$ and σ depend only on $\boldsymbol{D}$, which in turn depends only on $\boldsymbol{G}$. Letting $\boldsymbol{G} = \boldsymbol{LL^T}$ be the Cholesky decomposition of $\boldsymbol{G}$, and replacing (21) and (22) in (20) we get the profiled negative log likelihood as a function of $\boldsymbol{L}$ only (up to a constant):

$$\mathcal{L}(\boldsymbol{L} \mid \boldsymbol{y}) = \frac{N}{2}\log\big(\boldsymbol{y^TD^{-1}y} - \boldsymbol{y^TD^{-1}X(X^TD^{-1}X)^{-1}X^TD^{-1}y}\big) + \frac{1}{2}\log(\det(\boldsymbol{D})) \tag{23}$$

Parametrizing (23) with respect to the Cholesky decomposition $\boldsymbol{L}$ ensures that $\boldsymbol{G}$ remains positive definite within the iterative process of the minimizing algorithm. To express the gradient, we define the following variables: $\boldsymbol{\Lambda} := \boldsymbol{D(L)^{-1}}$, $\boldsymbol{C} := \boldsymbol{X(X^T\Lambda X)^{-1}X^T\Lambda}$, and $B := \boldsymbol{y^T\Lambda y} - \boldsymbol{y^T\Lambda Cy}$. Then the gradient is given by:

$$\frac{\mathrm{d}\mathcal{L}(\boldsymbol{L} \mid \boldsymbol{y})}{\mathrm{d}\,\mathrm{vech}(\boldsymbol{L})} = -2\boldsymbol{L_q}\,\mathrm{vec}(\boldsymbol{Z^T\Lambda}(\frac{N}{2B}(\boldsymbol{yy^T} - \boldsymbol{(Cyy^T + yy^TC^T)} + \boldsymbol{Cyy^TC^T}) - \frac{1}{2}\boldsymbol{D})\boldsymbol{\Lambda ZL}) \tag{24}$$

where $\boldsymbol{L_q}$ is an elimination matrix of order q and vech($\boldsymbol{L}$) is the vectorization of the lower triangular portion of $\boldsymbol{L}$.

For low dimensional problems, a numerical approximation of the gradient using finite differences might be a good alternative, but for high dimensional problems, the analytical gradient is a better option. The asymptotic complexity of the computation of the gradient is $\mathcal{O}(\max\{qN^2, pN^2\})$, provided that the matrix multiplication $\boldsymbol{\Lambda Z}$ is performed first, and the computation of $\boldsymbol{\Lambda}$ is done using Woodbury's matrix identiy $\boldsymbol{\Lambda} = \boldsymbol{I_N} - \boldsymbol{Z(G^{-1} + Z^TZ)^{-1}Z^T}$.

3.3.2.2 EM algorithm

We now consider the EM algorithm, and we derive analytical expressions for the updates of the parameters. Our goal is to minimize

$$Q(\boldsymbol{\beta}, \boldsymbol{G}, \sigma \mid \boldsymbol{\beta^{(i-1)}}, \boldsymbol{G^{(i-1)}}, \sigma^{(i-1)}) = E_{\mu \mid y, \beta^{(i-1)}, G^{(i-1)}, \sigma^{(i-1)}}[-\log(P(\boldsymbol{y}, \boldsymbol{\mu} \mid \boldsymbol{\beta}, \boldsymbol{G}, \sigma))] \tag{25}$$

Since $P(\boldsymbol{\mu} \mid \boldsymbol{y}, \boldsymbol{\beta^{(i-1)}}, \boldsymbol{G^{(i-1)}}, \sigma^{(i-1)}) \propto P(\boldsymbol{\mu}, \boldsymbol{y} \mid \boldsymbol{\beta^{(i-1)}}, \boldsymbol{G^{(i-1)}}, \sigma^{(i-1)})$, we can instead minimize the expectation with respect to the complete data likelihood. It is shown in (link to some reference) that:

$$P(\boldsymbol{\mu}, \boldsymbol{y} \mid \boldsymbol{\beta^{(i-1)}}, \boldsymbol{G^{(i-1)}}, \sigma^{(i-1)}) = \phi(\boldsymbol{y}, \boldsymbol{\beta^{(i-1)}}, \boldsymbol{G^{(i-1)}}, \sigma^{(i-1)}) pdf(\mathcal{N}(\boldsymbol{a^{(i-1)}}, {\sigma^2}^{(i-1)}(\boldsymbol{Z}^T\boldsymbol{Z} + {\boldsymbol{G}^{-1}}^{(i-1)})^{-1}))$$

where: $\boldsymbol{a^{(i-1)}} = (\boldsymbol{Z}^T\boldsymbol{Z} + {\boldsymbol{G}^{-1}}^{(i-1)})^{-1}\boldsymbol{Z}^T\boldsymbol{(y - X\beta^{(i-1)})}$ and $\phi(\boldsymbol{y}, \boldsymbol{\beta^{(i-1)}}, \boldsymbol{G^{(i-1)}}, \sigma^{(i-1)})$ is a constant that does not depend on $\boldsymbol{\mu}$

Therefore, we have (25) $\propto E_{\mu \sim \mathcal{N}(\boldsymbol{a^{(i-1)}}, {\sigma^2}^{(i-1)}(\boldsymbol{Z}^T\boldsymbol{Z} + {\boldsymbol{G}^{-1}}^{(i-1)})^{-1})}[-\log(P(\boldsymbol{y}, \boldsymbol{\mu} \mid \boldsymbol{\beta}, \boldsymbol{G}, \sigma))]$, which has the analytic form:

$$\begin{aligned}(N+q)\log(\sigma) + \frac{1}{2}\log(\det(\boldsymbol{G})) + \frac{1}{2\sigma^2}(\|\boldsymbol{y} - \boldsymbol{X\beta}\|_2^2 - 2(\boldsymbol{y} - \boldsymbol{X\beta})^T\boldsymbol{Za^{(i-1)}} \\ + \operatorname{Tr}\Big(({\sigma^2}^{(i-1)}(\boldsymbol{Z}^T\boldsymbol{Z} + {\boldsymbol{G}^{-1}}^{(i-1)})^{-1} + \boldsymbol{a^{(i-1)}}{\boldsymbol{a^{(i-1)}}}^T)(\boldsymbol{Z}^T\boldsymbol{Z} + \boldsymbol{G}^{-1})\Big))\end{aligned} \tag{26}$$

Minimizing with respect to $\boldsymbol{\beta}$, $\boldsymbol{G}$ and $\boldsymbol{\sigma}$ respectively, we get the update equations for the EM algorithm:

$$\boldsymbol{\beta^{(i)}} = (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{Za^{(i-1)}}) \tag{27}$$

$$\boldsymbol{G^{(i)}} = \frac{1}{{\sigma^2}^{(i-1)}}({\sigma^2}^{(i-1)}(\boldsymbol{Z}^T\boldsymbol{Z} + {\boldsymbol{G}^{-1}}^{(i-1)})^{-1} + \boldsymbol{a^{(i-1)}}{\boldsymbol{a^{(i-1)}}}^T) \tag{28}$$

$$\begin{aligned}\sigma^{(i)} = \{\frac{1}{N+q}(\|\boldsymbol{y} - \boldsymbol{X\beta^{(i)}}\|_2^2 - 2(\boldsymbol{y} - \boldsymbol{X\beta^{(i)}})^T\boldsymbol{Za^{(i-1)}} \\ + \operatorname{Tr}\Big[({\sigma^2}^{(i-1)}(\boldsymbol{Z}^T\boldsymbol{Z} + {\boldsymbol{G}^{-1}}^{(i-1)})^{-1} + \boldsymbol{a^{(i-1)}}{\boldsymbol{a^{(i-1)}}}^T)(\boldsymbol{Z}^T\boldsymbol{Z} + {\boldsymbol{G}^{-1}}^{(i)})\Big])\}^{1/2}\end{aligned} \tag{29}$$

Notice that the update for $\sigma^{(i)}$ depends on the updated values $\boldsymbol{\beta}^{(i)}$ and $\boldsymbol{G}^{(i)}$, and should therefore be performed after these updates are obtained. The asymptotic complexity of one iteration of the EM algorithm is $\mathcal{O}(\max\{qN, pN\})$. For large N, the EM algorithm can be much more efficient than first order and quasi-Newton methods per iteration. Methods have also been developed to accelerate the convergence of the EM algorithm such as SQUAR-EM [5], making it an efficient alternative to Quasi-Newton methods.

3.3.3 Generalized linear mixed models

The class of generalized linear mixed models extends the linear mixed model (18) by handling different types of observations. It is a class of mixed models with the following assumptions:

(a) $y_i \mid \boldsymbol{X_i}, \boldsymbol{\beta}, \boldsymbol{\mu} \in$ Natural exponential family (NEF) satisfying $E[y_i \mid \boldsymbol{X_i}, \boldsymbol{\beta}, \boldsymbol{\mu}] = g^{-1}(\boldsymbol{X_i\beta + Z_i\mu})$ for some link function $g : R \to R$.

(b) $\boldsymbol{\mu} \mid \boldsymbol{G} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{G})$

Letting $g(x) = id(x)$, we recover the linear mixed model (18). The other important link functions are $g(x) = \log(x)$ which leads to Poisson regression for count data, and $g(x) = \operatorname{logit}(x)$ which leads to logistic regression for binary data. As for the linear mixed model, the most commonly used methods of estimation are First-order/Quasi-Newton methods or the EM algorithm. As is well known, deriving the updates equations of these methods is significantly harder for GLMM's than it is for LMM's.

3.3.3.1 Standard methods

The goal is to maximize:

$$P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\beta}, \boldsymbol{G}) = \int_{\boldsymbol{\mu} \in \mathbb{R}^q} \prod_{i=1}^{N} P(y_i|\boldsymbol{X_i}, \boldsymbol{\beta}, \boldsymbol{\mu}) P(\boldsymbol{\mu}|\boldsymbol{G}) d\boldsymbol{\mu} \tag{30}$$

which does not have an analytical expression except for the case of an LMM. The difficulty in finding the maximum likelihood estimate resides in evaluating this integral accurately and efficiently. We give a short review of the methods used to perform this integration:

- **Laplace's Approximation:** Examining equation (30), we notice that the integrand, for constant $\boldsymbol{\beta}$ and $\boldsymbol{G}$, is proportional to the posterior of a logistic regression model in $\boldsymbol{\mu}$. By the Bernstein-Von mises theorem, this posterior is asymptotically Gaussian. Laplace's approximation works by replacing the integrand with its Gaussian approximation. The integral can then be computed analytically, and the function is maximized using standard first-order or Quasi-Newton methods. This approximation holds for large cluster sizes, and fails otherwise. More precisely, the leading error term is $\mathcal{O}(I^{-1})$ where I is the smallest cluster size (put reference here). In cases where the approximation holds, this method is very efficient.

- **Gauss-Hermite Quadrature:** This method approximates integrals of the form $\int_{-\infty}^{\infty} e^{-x^2} f(x) dx$ by a weighted sum of n-terms. It requires the computation of the roots of the n^{th} Hermite polynomial. If $f(x)$ is a polynomial of degree $\leq 2n-1$, this approximation is exact. Therefore, one way to evaluate the accuracy of this method is to see how well $f(x)$ is approximated by a polynomial of degree $2n-1$. This method can be generalized to the multivariate case in a straightforward manner, but as q grows larger, this method becomes rapidly unpractical since the total number of roots that have to be computed grows exponentially. An adaptive version of this method also exists where the maximum of the integrand is found first, which can increase the method's accuracy and efficiency.

- **Monte Carlo integration:** This is a large class of methods that approximates an arbitrary integral by casting it as an expectation, and using the central limit theorem to construct an estimator and determine its accuracy. Methods of this class typically require sampling from a specified distribution. In most cases, sampling from this distribution requires using MCMC methods.

An alternative to the direct maximization of (30) is to use the EM algorithm as in the LMM case. Here again, we are faced with evaluating the following intractable integral:

$$Q(\boldsymbol{\beta}, \boldsymbol{G} \mid \boldsymbol{\beta^{(i-1)}}, \boldsymbol{G^{(i-1)}}) = E_{\mu \mid y, \beta^{(i-1)}, G^{(i-1)}}[\log(P(\boldsymbol{y}, \boldsymbol{\mu} \mid \boldsymbol{\beta}, \boldsymbol{G}))] \tag{31}$$

The same methods mentioned above can be used to get an approximate answer. One expects the Gauss-Hermite Quadrature to perform better in evaluating (31) since $\log(P(\boldsymbol{y}, \boldsymbol{\mu} \mid \boldsymbol{\beta}, \boldsymbol{G}))$ can be approximated accurately using a polynomial of low order in many GLMM's. For a more formal discussion of the integration methods, the reader is referred to [4].

3.3.3.2 Applicability of our algorithm

Our algorithm can be used within the Monte Carlo integration framework. In the case of the integrals in (30) and (31), and when the clusters sizes are large, the distribution to sample from is approximately quadratic, and we can use our algorithm to perform this integration. This can offer an efficient alternative to the Laplace approximation method, much like in the hierarchical model case, since it can perform the optimization and sampling steps simultaneously.

4 Conclusion

In this research project we explored stochastic gradient MCMC methods. We showed how one can design such an algorithm in the general case, and used this design method to develop the new OSGFS algorithm for the special case of approximately quadratic log densities. We have also showed that our algorithm is optimal in terms of reduced autocorrelation and maximal rate of convergence among all continuous time MCMC algorithms. We have also developed a criterion that allows us to detect convergence of the Markov chain, removing the need of a burn-in period completely, and guaranteeing that all the samples collected are from the posterior. Finally, we verified the correctness of our algorithm by using it on standard statistical problems and showing that its results match those of standard approaches.

The main challenge is to generalize these ideas to arbitrary density functions. Unfortunately, many of the ideas we used in the development of the OSGFS algorithm rely on the fact that the target distribution is approximately Gaussian. In the general case, one needs to have access to the structure of the density to be able to develop such algorithms. One way to do that is to use a Taylor approximation of the target density. We have not yet explored this option, and we leave it for future investigations.

Appendix

In this appendix, we determine the error caused by the discretization of the process in the approximately quadratic log density case. We also derive the correct update equation (8) from the discretized process (7). We take this approach (instead of directly deriving the right discrete Markov process) to show how one can generalize this idea to other cases, where it is difficult to come up with the right discrete Markov process directly. In the last part of this appendix, we show that to minimize correlation between samples, and to maximize the rate of convergence of the chain, we must choose the largest possible k.

Discretization error

Consider the process described by (7), and remove the assumption that $\epsilon = 1$. Using the approximately quadratic log density assumption, and assuming, without loss of generality, that $\boldsymbol{\theta}_{MAP} = 0$ we can rewrite it as:

$$\boldsymbol{\theta}_{i+1} = (\boldsymbol{I} - \epsilon \boldsymbol{D}\boldsymbol{H})\boldsymbol{\theta}_i + \mathcal{N}(0, 2\epsilon \boldsymbol{D})$$

which is a VAR(1) process. This process has a stationary distribution if and only if:

$$-\boldsymbol{I} \prec (\boldsymbol{I} - \epsilon \boldsymbol{D}\boldsymbol{H}) \prec \boldsymbol{I}$$

If we let $\boldsymbol{D} = \frac{k}{2}\boldsymbol{H}^{-1}$ and $\epsilon := 1$, as we did in our discussion, the above condition is equivalent to $k \in (0, 1)$. As mentionned in our discussion in section 2.3.1.1, in the case of (14) with a strong prior, we can have $\boldsymbol{D}\boldsymbol{H} \succeq 2\boldsymbol{I}$, which makes the process non-stationary.

The stationary distribution of the process is $\mathcal{N}(0, \boldsymbol{\Lambda})$, where $\boldsymbol{\Lambda}$ satisfies:

$$\boldsymbol{\Lambda} = (\boldsymbol{I} - \epsilon \boldsymbol{D}\boldsymbol{H})\boldsymbol{\Lambda}(\boldsymbol{I} - \epsilon \boldsymbol{D}\boldsymbol{H}) + 2\epsilon \boldsymbol{D} \tag{32}$$

which has solution:

$$\boldsymbol{\Lambda} = 2\epsilon \sum_{i=0}^{\infty} (\boldsymbol{I} - \epsilon \boldsymbol{D}\boldsymbol{H})^i \boldsymbol{D} (\boldsymbol{I} - \epsilon \boldsymbol{D}\boldsymbol{H})^i$$

Letting $\boldsymbol{D} = \frac{k}{2}\boldsymbol{H}^{-1}$, we get:

$$\boldsymbol{\Lambda} = k\epsilon \sum_{i=0}^{\infty} (1 - \frac{k\epsilon}{2})^{2i} \boldsymbol{H}^{-1} = \frac{1}{1 - \frac{k\epsilon}{4}} \boldsymbol{H}^{-1}$$

Hence, the discretization error makes the process overestimate the true covariance. The ratio of the covariance of the stationary distribution of the process and the covariance of the target distribution approaches 1 linearly as $\epsilon \to 0$.

Derivation of the exact process

Let us restart from the process described by (7). Setting $\boldsymbol{\Lambda} = \boldsymbol{H}^{-1}$, we get from equation (32):

$$\boldsymbol{D}\boldsymbol{H}\boldsymbol{D} = 0$$

This term originates from the preconditioner of $\boldsymbol{\theta}_i$ in the update equation. This suggests that we should remove this term from the added noise to recover the desired process. Following this idea, we get the update equation :

$$\boldsymbol{\theta}_{i+1} = (\boldsymbol{I} - \boldsymbol{DH})\boldsymbol{\theta}_i + \mathcal{N}(0, 2\boldsymbol{D} - \boldsymbol{DHD})$$

which has the correct stationary distribution.

Maximizing the rate of convergence

The variance of the distribution at iteration t of the exact process, assuming $\boldsymbol{D} = \frac{k}{2}\boldsymbol{H}^{-1}$, is given by:

$$\mathrm{Var}[\boldsymbol{\theta}_t] = \boldsymbol{\Lambda}(t) = (k - \frac{k^2}{4})\sum_{i=0}^{t}(1 - \frac{k}{2})^{2i}\boldsymbol{H}^{-1} = [1 - (1 - \frac{k}{2})^t]\boldsymbol{H}^{-1}$$

We see that the rate of convergence is exponential in $(1 - \frac{k}{2})$, therefore maximizing k corresponds to maximizing the rate of convergence.

Minimizing auto-covariance

The autocovariance between two successive samples is given by:

$$\mathrm{Cov}[\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}] = E[(1 - \frac{k}{2})(\boldsymbol{\theta}_{t-1} + w_{t-1})^T\boldsymbol{\theta}_{t-1}] = (1 - \frac{k}{2}) = (1 - \frac{k}{2})\boldsymbol{\Lambda}$$

where w_{t-1} is the white noise added at time $t-1$. We conclude that the autocovariance between successive samples decreases linearly as $(1 - \frac{k}{2})$, and therefore to minimize it, k should be maximized.

References

[1] Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring. 2012.

[2] Yi-An Ma, Tianqi Chen, and Emily B. Fox. A Complete Recipe for Stochastic Gradient MCMC. pages 1–19, 2015.

[3] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic Gradient Descent as Approximate Bayesian Inference. 18:1–35, 2017.

[4] Francis Tuerlinckx, Frank Rijmen, Geert Verbeke, and Paul De Boeck. Statistical inference in generalized linear mixed models: A review. *British Journal of Mathematical and Statistical Psychology*, 59(2):225–255, 2006.

[5] Ravi Varadhan and Christophe Roland. Simple and globally convergent methods for accelerating the convergence of any em algorithm. *Scandinavian Journal of Statistics*, 35(2):335–353, 2008.