Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring

Paper Review by: David Carlson

Sungjin Ahn, Anoop Korattikara, and Max Welling
We begin with the simple question that this paper tries to answer:

*Can we approximately sample from a Bayesian posterior distribution if we are only allowed to touch a small mini-batch of data-items for every sample we generate?*
With a large dataset, MCMC methods become quite slow and get outperformed by simple methods based on stochastic gradients. To address this problem, Welling and Teh (2011) proposed a method called "Stochastic Gradient Langevin Dynamics" (SGLD) which combined ideas from Monte Carlo methods and stochastic gradients to approximately sample from the posterior at similar computational complexity to gradient descent (i.e. $O(n)$ to generate a sample).
Overview

With a large dataset, MCMC methods become quite slow and get outperformed by simple methods based on stochastic gradients. To address this problem, Welling and Teh (2011) proposed a method called “Stochastic Gradient Langevin Dynamics” (SGLD) which combined ideas from Monte Carlo methods and stochastic gradients to approximately sample from the posterior at similar computational complexity to gradient descent (i.e. $O(n)$ to generate a sample).

The problem was that this method mixed slowly due to very small stepsizes and could misestimate the posterior distribution. With large stepsizes a Metropolis-Hastings accept/reject step was necessary and ruined speed. The authors of this paper developed a method that can move quickly at large stepsizes and sample from the posterior accurately at small stepsizes (albeit with slow mixing rates).

Paper Review by: David Carlson
We first need to begin with some definitions of notation. We use the following definitions:

- $X_N$ is the collection of data $\{x_1, \ldots, x_N\}$ and a minibatch of $n$ samples is denoted $X_n^t$ for the $t^{th}$ minibatch.
- $\theta \in \Theta$ are the parameters of the problem and $\theta_0$ are the true generating parameters.
- $g_i(\theta) = \nabla_\theta \log p(x_i|\theta)$ and $\bar{g}_n(\theta; X_n^t) = \frac{1}{n} \sum_{i=1}^{n} g(\theta; x_{r_i})$ where $r$ indexes the minibatch.
- The Fisher information matrix is denoted $I(\theta) = \mathbb{E}_x[g(\theta; x)g(\theta; x)^T]$ and $I_N = NI(\theta)$.
- $I$ denotes the identity matrix.
The SGLD algorithm has the following update equation:

$$\theta_{t+1} \leftarrow \theta_t + \frac{\epsilon C}{2} \left\{ \nabla \log p(\theta_t) + N\bar{g}_n(\theta_t; X^t_n) \right\} + \nu$$

with $\nu \sim \mathcal{N}(0, \epsilon C)$

In this equation, $\epsilon$ is the step size, $C$ is the preconditioning matrix, and $\nu$ is a random variable representing injected Gaussian noise. Previous work shows that this update equation approximately sampled from the posterior. This method has the problems that the resulting Markov chain mixes slowly (but samples quickly!) for small values of $\epsilon$, where large values of $\epsilon$ require expensive Metropolis-Hastings accept/reject steps resulting in very slow steps.
Instead of naively using the SGLD algorithm, the authors suggest that we should use the fact that we know some asymptotic posterior distribution results to help us sample. Specifically, as the numbers of samples in a dataset $N$ grows large, the posterior distribution will become asymptotically Gaussian with $p(\theta | X_N) \approx \mathcal{N} \left( \theta_0, I_N^{-1} \right)$ according to the Bernstein-von Mises theorem.
Sampling from the approximate posterior

The SGLD has problems at large step sizes, but we can consider sampling from the Markov chain:

\[
\theta_{t+1} \leftarrow \theta_t + \frac{\epsilon}{2} C \{-I_N(\theta_t - \theta_0)\} + \omega
\]

where \( \omega \sim \mathcal{N}(0, \epsilon C - \frac{\epsilon^2}{4} CI_N C) \) (2)

If we manage to sample in this manner, then we can give the distributions of \( \theta_t \) and \( \theta_{t+1} \) by:

\[
\mu_{t+1} = (I - \frac{\epsilon}{2} CI_N) \mu_t + \frac{\epsilon}{2} CI_N \theta_0
\]

\[
\Sigma_{t+1} = (I - \frac{\epsilon}{2} CI_N) \Sigma_t (I - \frac{\epsilon}{2} CI_N)^T + \epsilon C - \frac{\epsilon^2}{4} CI_N C
\]

(3) (4)

Note that in this case we can show that the approximate posterior distribution \( \mathcal{N}(\theta_0, I_N^{-1}) \) is an \textit{invariant distribution} of this Markov chain. Therefore, one can sample from this chain at large step sizes without computing any MH accept-reject steps.
Motivation

The problem with SGLD is that the mixing can be so slow that we badly *misestimate* the posterior distribution because of very high sample correlation.

On the other hand, the new idea they propose is to sample from the approximate posterior approximated by a Gaussian distribution. However, if we wanted to find this approximation this is also an inefficient method because we could simply use the stochastic gradient to find the mean and the Hessian and approximate the posterior in that way without needing to gather samples.

In many cases, the approximate posterior will be inaccurate–the contribution of this paper is to combine the ideas of SGLD and the ideas of the approximate posterior so that the Markov chain can sample quickly yet also capture non-Gaussian effects in the posterior.
Motivation

The problem with SGLD is that the mixing can be so slow that we badly *misestimate* the posterior distribution because of very high sample correlation.

On the other hand, the new idea they propose is to sample from the approximate posterior approximated by a Gaussian distribution. However, if we wanted to find this approximation this is also an inefficient method because we could simply use the stochastic gradient to find the mean and the Hessian and approximate the posterior in that way without needing to gather samples.
Motivation

The problem with SGLD is that the mixing can be so slow that we badly misestimate the posterior distribution because of very high sample correlation.

On the other hand, the new idea they propose is to sample from the approximate posterior approximated by a Gaussian distribution. However, if we wanted to find this approximation this is also an inefficient method because we could simply use the stochastic gradient to find the mean and the Hessian and approximate the posterior in that way without needing to gather samples.

In many cases, the approximate posterior will be inaccurate—the contribution of this paper is to combine the ideas of SGLD and the ideas of the approximate posterior so that the Markov chain can sample quickly yet also capture non-Gaussian effects in the posterior.

Paper Review by: David Carlson
The authors introduce a new algorithm, Stochastic Gradient Fisher Scoring (SGFS), to combine properties from SGLD discussed earlier as well as the discussion on sampling from the approximate posterior

\[
\theta_{t+1} \leftarrow \theta_t + \frac{\epsilon}{2} \mathbf{C} \{ \nabla \log p(\theta_t) + N\bar{g}(\theta_t; X^t_n) \} + \tau \tag{5}
\]

where \( \tau \sim \mathcal{N}(0, \mathbf{Q}) \)

\( \mathbf{Q} \) is a positive definite matrix that will be discussed later. If \( n \) is large enough, we have that:

\[
\bar{g}_n(\theta_t; X^t_n) \sim \mathcal{N}(\mathbb{E}_x[g_n(\theta_t; x)], \frac{1}{n} \text{Cov}[g(\theta_t; x)]) \tag{6}
\]

We are now ready to derive properties of the sampler.
To begin, consider the interior part of Equation 5, letting $G_N(\theta_t; X_n) = \nabla_{\theta_t} p(X_n|\theta_t)$:

$$\nabla \log p(\theta_t) + N\bar{g}_n(\theta_t; X_n^t) \approx \nabla \log p(\theta_t) + G_N(\theta_t; X_n) + \phi$$  \hspace{1cm} (7)

where $\phi \sim \mathcal{N}\left(0, \frac{NI_N^{-1}}{n}\right)$

If the posterior is close to its Bernstein-von Mises approximation, we have:

$$\nabla \log p(\theta_t) + G_N(\theta_t; X_n) = -I_N(\theta_t - \theta_0)$$  \hspace{1cm} (8)

Note this is the same term that we find in Equation 2 where we were sampling from the approximate posterior!
SGFS Properties

We can then rewrite the SGFS equation as:

$$\theta_{t+1} \leftarrow \theta_t + \frac{\epsilon}{2} \mathbf{C}\{-\mathbf{I}_N(\theta_t - \theta_0)\} + \psi + \tau$$

with $\psi \sim \mathcal{N}\left(0, \frac{\epsilon^2 N}{4n} \mathbf{C}\mathbf{I}_N^{-1}\mathbf{C}\right)$ and $\tau \sim \mathcal{N}(0, \mathbf{Q})$

We can show that we should set:

$$\mathbf{Q} = \begin{cases} 
\epsilon \mathbf{C} & \text{for small } \epsilon \\
\epsilon \mathbf{C} - \epsilon^2 (N + n) \mathbf{C}\mathbf{I}_N \mathbf{C}/(4n) & \text{for large } \epsilon
\end{cases}$$

(10)
Choosing C

We need to choose $C$ so that the covariance matrix of the injected noise is positive-definite. This can be enforced by setting:

$$
\epsilon C - \frac{\epsilon^2}{4} \frac{N + n}{n} CI_N C = \epsilon CBC
$$

where $B$ is any symmetric positive definite matrix; it is recommended to set $B \propto I_N$. $I_N$ is approximated as $\hat{N}_1, t$, which is computed by an online average with:

$$
\hat{I}_{1, t} = (1 - \kappa_t)\hat{I}_{1, t-1} + \kappa_t V(\theta_t; X^t)
$$

where $\kappa_t = 1/t$. 

Paper Review by: David Carlson
Algorithm 1: Stochastic Gradient Fisher Scoring (SGFS)

**Input:** \( n, B, \{\kappa_t\}_{t=1:T} \)

**Output:** \( \{\theta_t\}_{t=1:T} \)

1: Initialize \( \theta_1, \hat{I}_{1,0} \)

2: \( \gamma \leftarrow \frac{n+N}{n} \)

3: **for** \( t = 1 : T \) **do**

4: Choose random minibatch \( X^t_n = \{x_{t_1} \ldots x_{t_n}\} \)

5: \( \overline{g}_n(\theta_t) \leftarrow \frac{1}{n} \sum_{i=1}^{n} g_{t_i}(\theta_t) \)

6: \( V(\theta_t) \leftarrow \frac{1}{n-1} \sum_{i=1}^{n} \{g_{t_i}(\theta_t) - \overline{g}_n(\theta_t)\} \{g_{t_i}(\theta_t) - \overline{g}_n(\theta_t)\}^T \)

7: \( \hat{I}_{1,t} \leftarrow (1 - \kappa_t)\hat{I}_{1,t-1} + \kappa_t V(\theta_t) \)

8: Draw \( \eta \sim \mathcal{N}[0, \frac{4B}{\varepsilon}] \)

9: \( \theta_{t+1} \leftarrow \theta_t + 2 \left( \gamma N\hat{I}_{1,t} + \frac{4B}{\varepsilon} \right)^{-1} \{\nabla \log p(\theta_t) + N\overline{g}_n(\theta_t) + \eta\} \)

10: **end for**
Experiments

Several experiments were run:

- Logistic regression on the digits 7 and 9 using a 50-dimensional random projection and used 3,000 burn-in and 100,000 samples.
- 3 layer neural network with logistic activation functions on the Heritage Health Prize and the MNIST dataset for 10 digit classification.
- Discriminative Restricted Boltzmann Machine on the KDD99 dataset.

\( ATUC \) is defined as \( \text{time per sample} \times (1 + 2 \sum_{s=1}^{\infty} \rho(s)) \)

where \( \rho(s) \) is the autocorrelation at lag \( s \).
Figure 1. 2-d marginal posterior distributions for logistic regression. Grey colors correspond to samples from SGFS. Red solid and blue dotted ellipses represent iso-probability contours at two standard deviations away from the mean computed from HMC and SGFS, respectively. Top plots are the results for SGFS-f and bottom plots represent SGFS-d. Plots on the left represent the 2-d marginals with the smallest difference between HMC and SGFS while the plots on the right represent the 2-d marginals with the largest difference. Value for $\alpha$ is 0 meaning that no additional noise was added.
Experiments

Figure 2. Final error of logistic regression at time T versus mixing rate for the mean (top) and covariance (bottom) estimates after 100 (left) and 3000 (right) seconds of computation. See main text for detailed explanation.
Experiments

**Figure 3.** Test-set classification error of NNs trained with SGFS-f, SGFS-d, SGLD and SGD on the HHP dataset (left) and the MNIST dataset (right).
Figure 4. 2-d marginal posterior distributions of DRBM. Grey colors correspond to samples from SGFS/SGLD. Thick red solid lines correspond to iso-probability contours at two standard deviations away from the mean computed from HMC samples. Thin red solid lines correspond to HMC results based on subsets of the samples. The thick blue dashed lines correspond to SGFS-f (top) and SGLD (bottom) runs. Plots on the left represent the 2-d marginals with the smallest difference between HMC and SGFS/SGLD while the plots on the right represent the 2-d marginals with the largest difference.
Experiments

Figure 5. Final error for DRBM at time T versus mixing rate for the mean (left) and covariance (right) estimates after 6790 seconds of computation on a subset of KDD99.

<table>
<thead>
<tr>
<th>Method</th>
<th>Rel. Err. in Mean at 6790 sec.</th>
<th>Rel. Err. in Cov at 6790 sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>$8.010^{-4}$</td>
<td></td>
</tr>
<tr>
<td>SGLD</td>
<td>$6.610^{-4}$</td>
<td></td>
</tr>
<tr>
<td>SGFS-d</td>
<td>$4.210^{-4}$</td>
<td></td>
</tr>
<tr>
<td>SGFS-f</td>
<td>$4.410^{-4}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Final test error rate on the KDD99 dataset.