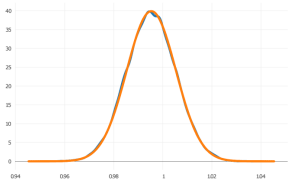
Experiments with Stochastic gradient descent MCMC

Ayoub El Hanchi

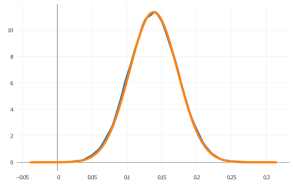
October 25, 2018

1 One dimensional toy problems

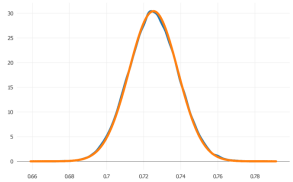
To start with, I ran OSGFS (See last page for the update equation) on some basic one dimensional posteriors from 10 000 simulated samples of data, and checked its behavior as I varied the batch size, and as the posterior deviates from a Gaussian. S below refers to the batch size.



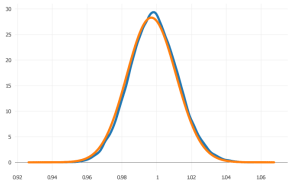
(a) Gaussian, S = 100



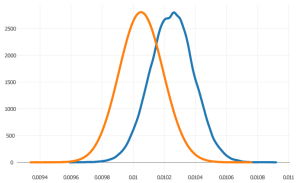
(b) Logistic, S = 100



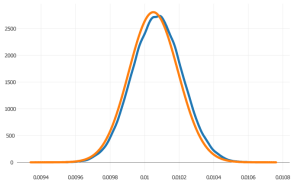
(c) Poisson, S = 100



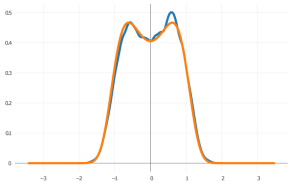
(c) Gamma, mode = 1, S = 100



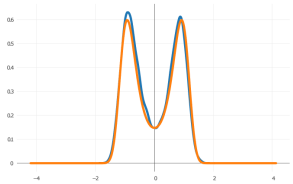
(d) Gamma, mode = 0.01, S = 100



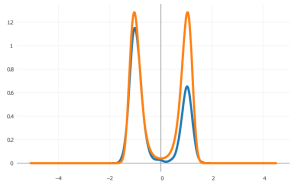
(e) Gamma, mode = 0.01, S = 1000



(f) Gaussian mixture, S = 100



(g) Gaussian mixture, S = 100



(h) Gaussian mixture, S = 100

I noted three main points when doing these experiments:

1. The first three experiments show that OSGFS performs well when the posterior is close to a Gaussian, as expected.
2. From the experiments on the Gaussian mixtures, we can see that the algorithm can properly sample from multi-modal distributions, provided that the region separating the modes has a minimal amount of density.
3. The case of a Gamma posterior is intriguing. The algorithm performs differently depending on the choice of the batch size. In both cases, the true posterior is very close to a Gaussian, but for some unknown reason, when using a smaller batch size, the algorithm converges to a shifted version of the posterior, preserving the right variance. I encountered the same problem when doing large scale logistic regression. I will discuss this more in the next section.

2 Large scale Logistic regression

2.1 Approximately Gaussian posterior

As a second step, I used OSGFS to fit a logistic regression model on the *Covertype* data set which has 581012 observations and 54 attributes from the UCI Machine learning repository. The posterior in this case is very close to a Gaussian since the observed fisher information matrix at the minimum matches the Hessian at the minimum with a significant degree of accuracy.

I used the R package "sgmcmc", which was developed by the authors of the stochastic gradient Langevin dynamics (SGLD) paper, to sample from the posterior using SGLD (one of the vignettes posted with the package is on logistic regression applied to this data set, I followed the vignette). I also used OSGFS with different batch sizes, and compared the output of each with the Laplace approximation of the posterior. The results are show in figure (1) and (2).

As can be seen from figure (2), the same phenomenon observed in the case of a unidimensional gamma posterior happens here. When the batch size is much smaller than the size of the data set ($< 1\%$), the algorithm converges to a shifted version of the posterior, again preserving the covariance structure. Nonetheless, the algorithm performs well for $S \geq 5000$, or $\geq 1\%$ of the original data set size. From figure (1), we see that the chain constructed by OSGFS suffers much less from autocorrelation than the chain constructed by SGLD. Note that the number of samples collected using different batch sizes is proportional to the size of the batch size, and that the number of iterations appearing in figure (1) was adjusted so that the same amount of data has been processed by the different versions of the algorithm after each iteration.

In terms of computation time, it took my machine ≈ 1 minute to get 10 000 samples with a batch size of 5000 using OSGFS, and a similar time using SGLD with the same batch size, using the implementation of the package "sgmcmc". In general, SGLD as presented in the original paper is less computationally expensive given that it does not require a precondtioner in its update equation, whereas OSGFS does.

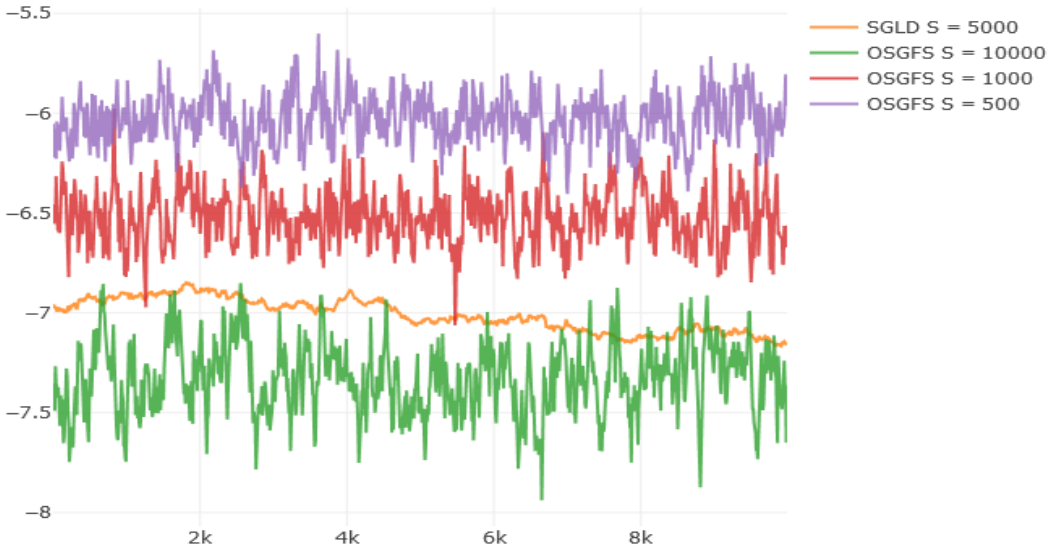


Figure 1: First component of the parameter vector as a function of the number of iterations

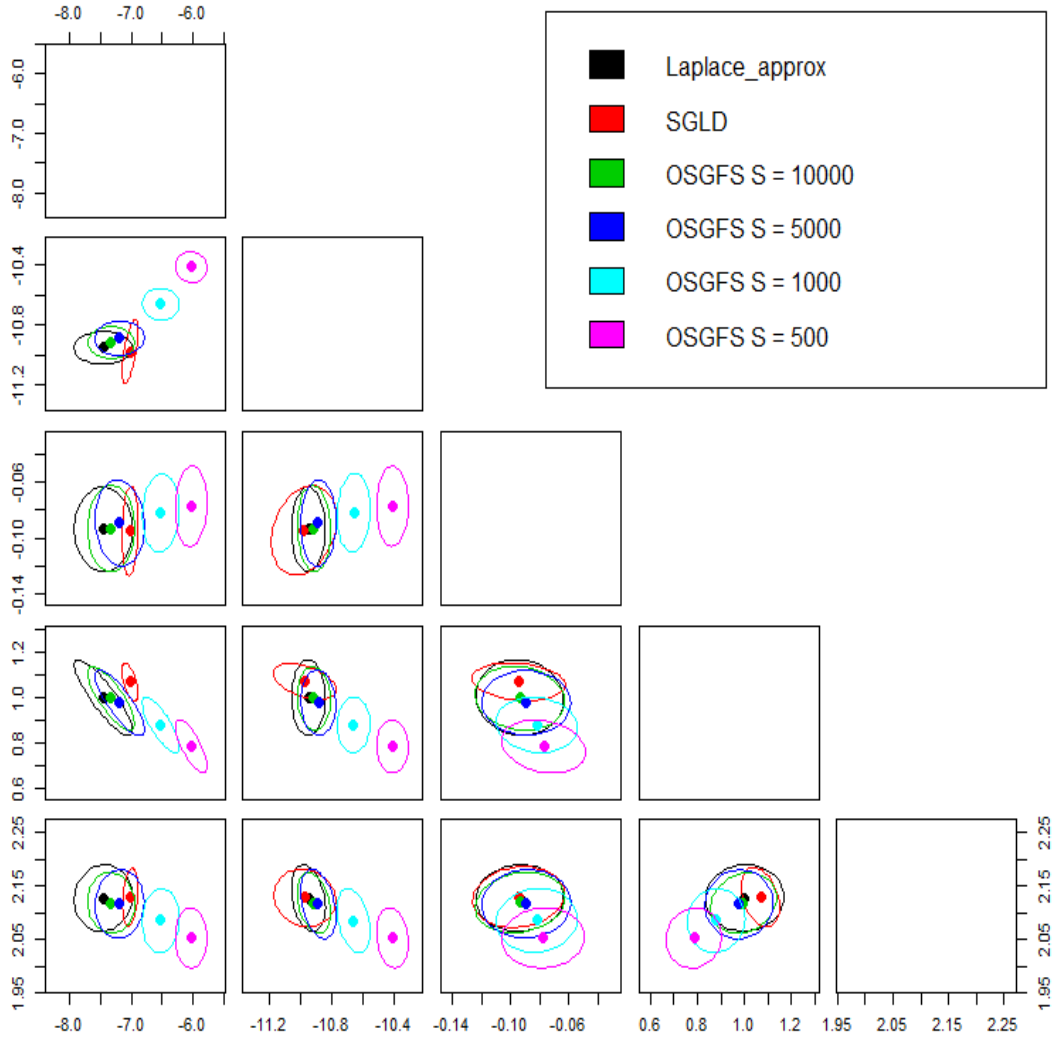


Figure 2: Mean and covariance structure of 5 randomly chosen components of the parameter vector of logistic regression on the *Covertype* data set

2.2 Non Gaussian posterior

To test the algorithm on a non Gaussian logistic regression posterior, I used the *MNIST* data set, and applied logistic regression on the digits 1 and 2. After taking the first 30 principal components of the data set, its final size was 11623 observations with 30 attributes. This resulted in a highly non Gaussian posterior.

In order to assess the performance of OSGFS, I used Hamiltonian Monte carlo (HMC) to get 10000 samples from the posterior using a step size of $\epsilon = 0.01$ and $L = 10$ leapfrog steps. I then used OSGFS with batch sizes $S = 500$ and $S = 2000$. I have also attempted to use SGLD on this problem, with different step size values using the package "sgmcmc", but the function always returned with an error saying the chain diverged.

In figure (4), we see that the mean and covariance of the distribution predicted by OSGFS match very well the ones predicted by HMC. Looking at figure (3), we see that OSGFS is able to adapt to the non Gaussian structure of the posterior, and samples properly from the regions of high density far from the mode. With the same amount of samples, samples from HMC suffer less from autocorrelation than the ones from OSGFS. On the other hand, to get 10 000 samples using HMC took $\approx$ 15 minutes on my machine, whereas OSGFS took $\approx$ 10 seconds for the same amount of samples. On the scale of independent samples per computational time unit, OSGFS outperforms HMC in this case. I did not encounter the problem of the "shifted" posterior shown in the previous section in this case since the batch sizes were large enough to avoid the effect.

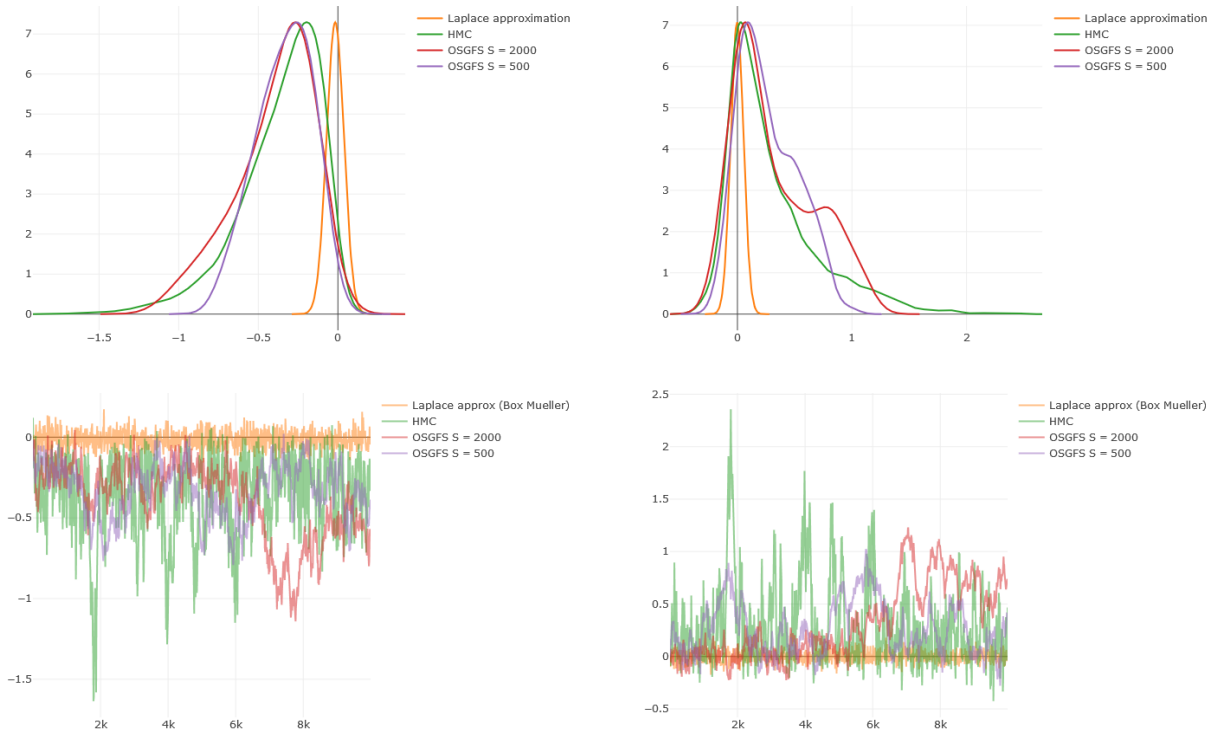


Figure 3: Density estimation of two highly non gaussian components of the posterior using Hamiltonian Monte Carlo and OSGFS.

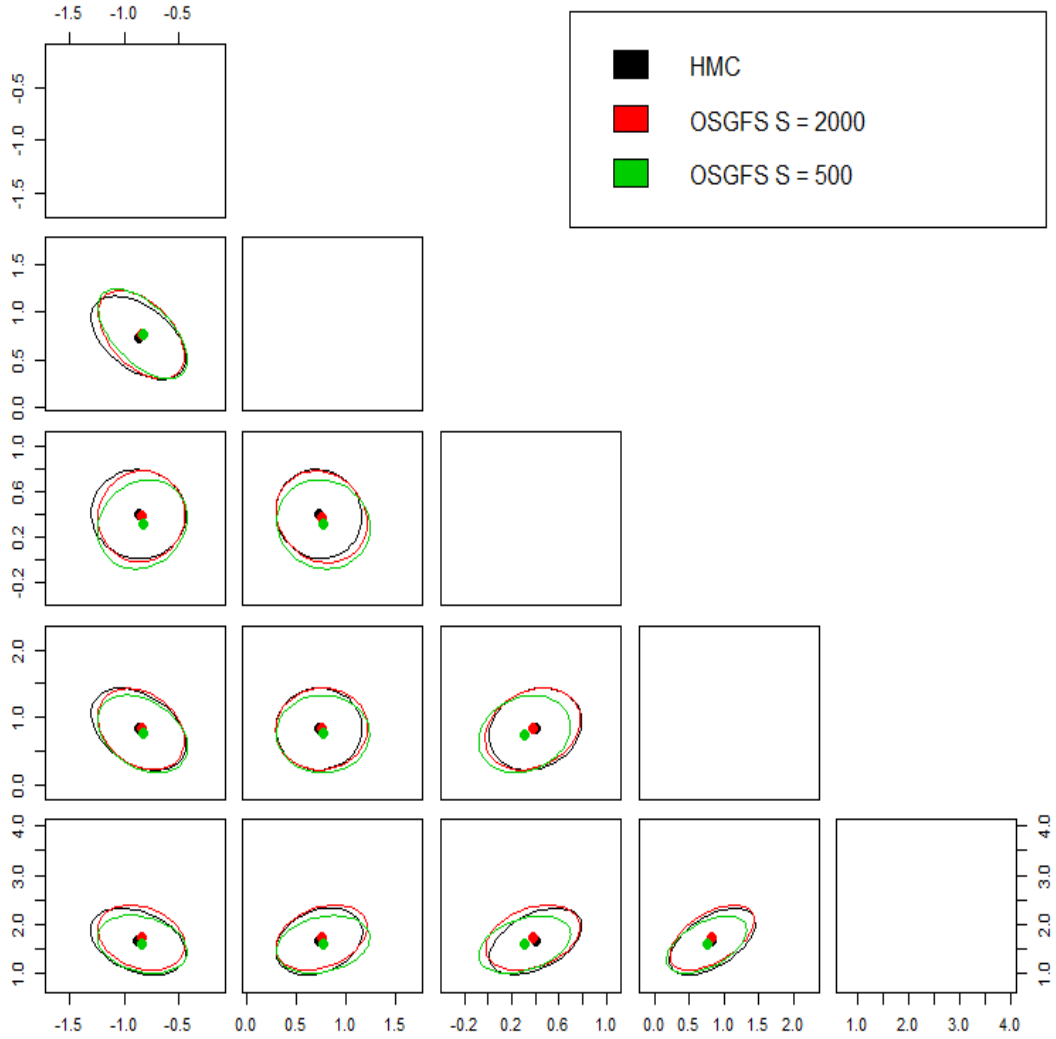


Figure 4: Mean and covariance of 5 randomly chosen components of the parameter vector of logistic regression on the *MNIST* data set

3 Neural network

I built a neural network of one hidden layer with 10 hidden units. I used hyperbolic tangent as the activation function for the hidden units and the logistic sigmoid for the activation function of the output unit. I used this neural network on the modified MNIST data set as described in the previous section to do binary classification.

I applied OSGFS and SGHMC to train the network, as well as regular SGD, and SGD with momentum. Due to the highly non quadratic and the large regions of almost zero curvature in the log likelihood, both sampling algorithms performed worse that the optimization algorithms. I have tried the same algorithms on 1 unit, hoping to increase the curvature of the log likelihood, but the results remained the same. I have also tried to first find a local minimum using SGD with momentum, and then apply OSGFS, but this failed as well since all the local minima I found had very low curvature around them.

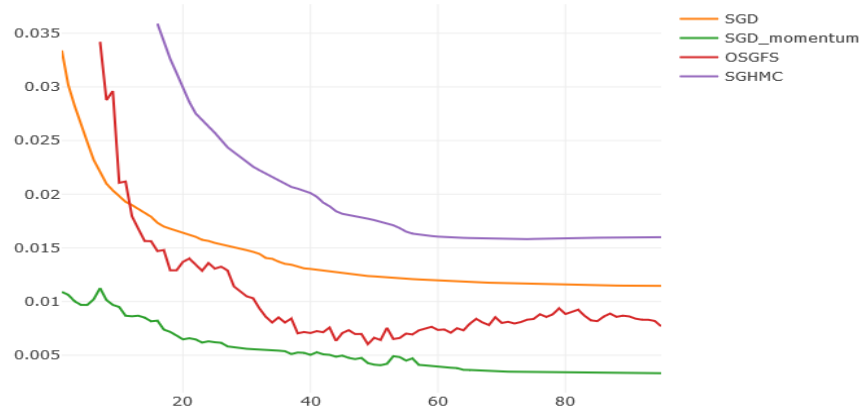


Figure 5: Negative Log loss of the training set reached by each algorithm as a function of the number of iterations (number of iterations is by hundreds)

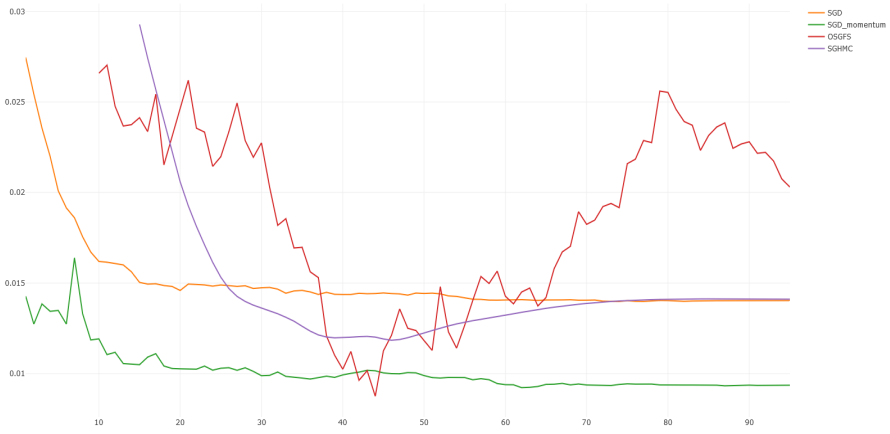


Figure 6: Negative Log loss of the test set reached by each algorithm as a function of the number of iterations (number of iterations is by hundreds)

4 Update equation

Throughout the text, I used the acronym OSGFS to refer to the algorithm with the following update equation:

$$\Delta\theta = -\epsilon \boldsymbol{D} \nabla_\theta^S \log f(\theta) + \mathcal{N}(0, 2\epsilon \boldsymbol{D} - \epsilon^2 \frac{N^2}{S} \boldsymbol{D}\boldsymbol{C}\boldsymbol{D} - \epsilon^2 \boldsymbol{D}\boldsymbol{H}\boldsymbol{D}) \tag{1}$$

where $\boldsymbol{D} = 2\frac{S}{N+S}\boldsymbol{H}^{-1}$, with $\boldsymbol{H}$ being the Hessian at the mode, $\boldsymbol{C}$ is an estimate of the covariance of the gradients near the mode, $f(\theta)$ is the unormalized posterior density and ∇_θ^S refers to the stochastic gradient computed with a batch size S. For problems where the posterior is almost Gaussian, $\epsilon = 1$ is used, while for problems where the posterior deviates significantly from a Gaussian, ϵ is maximized with the constraint that the noise covariance is positive semi-definite (around 0.1 in the problems I tried).