Stochastic Newton Sampler: R Package sns

Alireza S. Mahani
Scientific Computing
Sentrana Inc.

Asad Hasan
Scientific Computing
Sentrana Inc.

Marshall Jiang
Department of Mathematics
Cornell University

Mansour T.A. Sharabiani
National Heart and Lung Institute
Imperial College London

Abstract

The R package **sns** implements Stochastic Newton Sampler (SNS), a Metropolis-Hastings Monte Carlo Markov Chain algorithm where the proposal density function is a multivariate Gaussian based on a local, second-order Taylor series expansion of log-density. The mean of the proposal function is the full Newton step in Newton-Raphson optimization algorithm. Taking advantage of the local, multivariate geometry captured in log-density Hessian allows SNS to be more efficient than univariate samplers, approaching independent sampling as the density function increasingly resembles a multivariate Gaussian. SNS requires the log-density Hessian to be negative-definite everywhere in order to construct a valid proposal function. This property holds, or can be easily checked, for many GLM-like models. When initial point is far from density peak, running SNS in non-stochastic mode, i.e., taking the Newton step with line search, allows the MCMC chain to converge to high-density areas faster. For high-dimensional problems, partitioning of state space into lower-dimensional subsets, and applying SNS to the subsets within a Gibbs sampling framework can significantly improve the mixing of SNS chains. In addition to the above strategies for improving convergence and mixing, **sns** offers a rich set of diagnostics and visualization capabilities, as well as a function for sample-based calculation of Bayesian predictive posterior distributions.

Keywords: monte carlo markov chain, metropolis-hastings, newton-raphson optimization, negative-definite hessian, log-concavity.

1. Introduction

In most real-world applications of Monte Carlo Markov Chain (MCMC) sampling, the probability density function (PDF) being sampled is multidimensional. Univariate samplers such as Slice Sampler (Neal 2003) and Adaptive Rejection Sampler (Gilks and Wild 1992) can be embedded in the Gibbs sampling framework (Geman and Geman 1984) to sample from multivariate PDFs (Mahani and Sharabiani 2015b). Univariate samplers generally have few tuning parameters, making them ideal candidates for black-box MCMC software such as JAGS (Plummer 2013) and OpenBUGS (Thomas, O'Hara, Ligges, and Sturtz 2006). However, they become less effective as PDF dimensionality rises and dimensions become more correlated (Girolami and Calderhead 2011). Therefore, development - and software implementation - of efficient, black-box multivariate MCMC algorithms is of great importance to widespread application of probabilistic models in statistics and machine learning.

The R package **sns** implements Stochastic Newton Sampler (SNS), a Metropolis-Hastings MCMC algorithm (Hastings 1970), where the proposal distribution is a locally-fitted multivariate Gaussian resulting from second-order Taylor series expansion of the log-density. In its current implementation, SNS requires the log-density to be twice-diffenrentiable and globally concave, or equivalently that its Hessian matrix be negative-definite everywhere. For many Generalized Linear Models (GLMs) these conditions are satisfied (Gilks and Wild 1992), and the invariance theorem of Mahani and Sharabiani (2015a) allows Hessian negative-definiteness to be studied and proven in the much lower-dimensional space of base distributions, rather than the high-dimensional space of regression coefficients.

SNS has appeared in the literature under several variations and labels. Gamerman (1997) extend Iterative Reweighted Least Squares (IRLS) - the primary estimation technique for GLM models - to MCMC sampling by adding a Metropolis-Hastings step to it. Given that IRLS is a close cousin of Newton-Raphson optimization, their method can be considered a specialization of SNS for GLM models. Qi and Minka (2002) present what they call 'Hessian-based Metropolis-Hastings' (HMH), which is nearly identical to SNS, but they do not address the high-dimensional mixing problem, nor do they provide an open-source software implementation. More recently, the simplified manifold Metropolis adjusted Langevin Algorithm (MMALA) of Girolami and Calderhead (2011) is very similar to SNS with the addition a tunable step size, or learning rate. The software accompanying their paper is written in MATLAB (MATLAB 2014). The R package **sns**, to our knowledge, is the first open-source implemnentation of the SNS algorithm, including extensions for improving convergence (`rnd` and`nnr` arguments) and mixing (`part` argument), diagnostic and visualization methods (`summary.sns` and `plot.sns`), and sample-based prediction (`predict.sns`).

The paper is organized as follows. In Section 2, we review the theoretical background for SNS, including an overview of Metropolis-Hastings algorithms, followed by the multivariate Gaussian proposal PDF used in SNS. In Section 3, we discuss the implementation of SNS algorithm in the **sns** package. Section 4 offers several examples to illustrate the usage of **sns**. Finally, Section 5 provides a summary and pointers for future extensions to the software as well as potential research directions.

2. Theory

We begin with a brief overview of the Metropolis-Hastings algorithm.

2.1. Metropolis-Hastings algorithm

In Metropolis-Hastings (MH) MCMC sampling of the PDF, $p(\mathbf{z})$, we generate a sample $\mathbf{z}^*$ from the proposal density function $q(\mathbf{z}|\mathbf{z}^\tau)$, where $\mathbf{z}^\tau$ is the current state. We then accept the proposed state $\mathbf{z}^*$ with probability $A(\mathbf{z}^*, \mathbf{z}^\tau)$, where:

$$A(\mathbf{z}^*, \mathbf{z}^\tau) = \min\left(1\,,\, \frac{p(\mathbf{z}^*)q(\mathbf{z}^\tau|\mathbf{z}^*)}{p(\mathbf{z}^\tau)q(\mathbf{z}^*|\mathbf{z}^\tau)}\right) \tag{1}$$

The MH transitions satisfy detailed balance:

$$\begin{aligned} p(\mathbf{z})q(\mathbf{z}|\mathbf{z}')A(\mathbf{z}',\mathbf{z}) &= \min(p(\mathbf{z})q(\mathbf{z}|\mathbf{z}'), p(\mathbf{z}')q(\mathbf{z}'|\mathbf{z})) \\ &= \min(p(\mathbf{z}')q(\mathbf{z}'|\mathbf{z}), p(\mathbf{z})q(\mathbf{z}|\mathbf{z}')) \\ &= p(\mathbf{z}')q(\mathbf{z}'|\mathbf{z})A(\mathbf{z},\mathbf{z}') \end{aligned} \tag{2}$$

The detailed balance property ensures that $p(\mathbf{z})$ is invariant under MH transitions. For a discussion of ergodicity of MH algorithm, see Roberts and Rosenthal (1999).

2.2. SNS proposal density

SNS proposal density is a multivariate Gaussian fitted locally to the density being sampled, using the second-order Taylor-series expansion of the log-density:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \mathbf{g}(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top \mathbf{H}(\mathbf{x}_0)\,(\mathbf{x} - \mathbf{x}_0) \tag{3}$$

where $f : \mathbb{R}^K \to \mathbb{R}$, and $\mathbf{g}$ and $\mathbf{H}$ are the gradient vector and Hessian matrix for f, respectively, of dimensions K and $K \times K$. Assuming that f is globally concave, the above approximation is equivalent to fitting the following multivariate Gaussian (which we refer to as $F(\mathbf{x})$) to the PDF:

$$F(\mathbf{x}) = \frac{1}{(2\pi)^{K/2}|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\} \tag{4}$$

By comparing Equations 3 and 4, we see that the precision matrix is the same as negative Hessian: $\mathbf{\Sigma}^{-1} = -\mathbf{H}(\mathbf{x}_0)$. The mean of the fitted Gaussian maximizes its log, and therefore:

$$\boldsymbol{\mu} = \mathbf{x}_0 - \mathbf{H}^{-1}(\mathbf{x}_0)\,\mathbf{g}(\mathbf{x}_0) \tag{5}$$

We can now formally define the (multivariate Gaussian) proposal density $q(.\,|\,\mathbf{x})$ as:

$$q(.\,|\,\mathbf{x}) = \mathcal{N}(\mathbf{x} - \mathbf{H}^{-1}(\mathbf{x})\,\mathbf{g}(\mathbf{x})\,,\, -\mathbf{H}^{-1}(\mathbf{x})) \tag{6}$$

Note that Equation 5 is simply the full Newton step (Nocedal and Wright 2006a). We can therefore think of SNS as the stochastic counterpart of Newton-Raphson (NR) optimization. In NR optimization, we select the mean of the fitted Gaussian as the next step, while in SNS we draw a sample from the fitted Gaussian and apply MH test to accept or reject it. Also, note that in the special case where the sampled PDF is Gaussian, $f(\mathbf{x})$ is quadratic and therefore the proposal function is identical to the sampled PDF. In this case $A(\mathbf{z}', \mathbf{z})$ is always equal to 1, implying an acceptance rate of 100%.

3. Software implementation and features

3.1. Overview

The workhorse of **sns** package is the `sns` function, responsible for implementation of MH algorithm using the multivariate Gaussian proposal density described in Section 2.2. `sns` implements the following steps:

1. Evaluate the log-density function and its gradient and Hessian at $\mathbf{x}_{\text{old}}$: $f_{\text{old}}, \mathbf{g}_{\text{old}}, \mathbf{H}_{\text{old}}$.

2. Construct the multivariate Gaussian proposal function at $q(.|\mathbf{x}_{old})$ using Equation 6 and $\mathbf{x} = \mathbf{x}_{old}$.

3. Draw a sample $\mathbf{x}_{prop}$ from $q(.|\mathbf{x}_{old})$, and evaluate $logq_{prop} = \log(q(\mathbf{x}_{prop}|\mathbf{x}_{old}))$.

4. Evaluate the log-density function and its gradient and Hessian at $\mathbf{x}_{prop}$: $f_{prop}, \mathbf{g}_{prop}, \mathbf{H}_{prop}$.

5. Construct the multivariate Gaussian proposal function at $q(.|\mathbf{x}_{prop})$ using Equation 6 and $\mathbf{x} = \mathbf{x}_{prop}$, and evaluate $logq_{old} = \log(q(\mathbf{x}_{old}|\mathbf{x}_{prop}))$.

6. Calculate the ratio $r = \exp((f_{prop} - f_{old}) + (logq_{old} - logq_{prop}))$.

7. If $r \geq 1$ accept $\mathbf{x}_{prop}$: $\mathbf{x}_{new} \leftarrow \mathbf{x}_{prop}$. Else, draw a random deviate s from a uniform distribution over $[0, 1)$. If $s < r$, then accept $\mathbf{x}_{prop}$: $\mathbf{x}_{new} \leftarrow \mathbf{x}_{prop}$, else reject $\mathbf{x}_{prop}$: $\mathbf{x}_{new} \leftarrow \mathbf{x}_{old}$.

Fitting the multivariate Gaussian in steps 2 and 5 is done via calls to the private function `fitGaussian`. We use the functions `dmvnorm` and `rmvnorm` from package **mvtnorm** to calculate the log-density of, and draw samples from, multivariate Gaussian proposal functions.

There are two important arguments in `sns`, namely `rnd` and `part`. The first argument, `rnd`, controls whether the algorithm should run in stochastic or MCMC mode (which is the default choice), or in non-stochastic or Newton-Raphson (NR) mode. The second argument, `part`, controls the state space partitioning strategy. These arguments and their roles are described in Section 3.2.

`sns.run` is a wrapper around `sns`, and offers the following functionalities:

1. Convenience of generating multiple samples via repeated calls to `sns`. After the first call, the Gaussian fit object (attached as attribute `gfit` in the returned value from `sns`) is fed by `sns.run` back to `sns` via argument `gfit`, in order to avoid unnecessary fitting of the proposal function at current value.

2. Collecting diagnostic information such as log-probability (time series), acceptance rate, relative deviation from quadratic approximation (time series), and components of MH test. These diagnostic measures are discussed in Section 3.3, and their use is illustrated via examples in Section 4.

The generic methods `summary.sns`, `plot.sns` and `predict.sns` provide diagnostic, visualization, and prediction capabilities, discussed in Sections 3.3 and 3.4, and illustrated via examples in Section 4.

3.2. Improving convergence and mixing

NR mode: Far from the distribution mode, the local multivariate Gaussian fit can be severely different from the PDF, leading to small overlap between the two, low acceptance rate and hence bad convergence. This can be overcome by spending the first few iterations in non-stochastic or NR mode, where instead of drawing from the proposal function we simply accept its mean as the next step. Rather than taking a full Newton step, we have implemented line search (Nocedal and Wright 2006b) to ensure convergence to the PDF maximum. To use `sns` in NR mode, users can set the argument `rnd` to `FALSE`. In NR mode, each iteration is guaranteed to increase the log-density. Using the NR mode during the initial burn-in phase is illustrated in Section 4. In `sns.run`, the argument `nnr` controls how many initial iterations to be performend in NR mode.

State space partitioning: Even when near the PDF maximum, the fitted Gaussian can be severely different from the PDF. This can happen if the PDF has a significant third derivative, a phenomenon that we have observed for high-dimensional problems, especially when the number of observations is small. To improve bad mixing in high dimensions, we use a strategy which we refer to as 'state space partitioning', where state space is partitioned into disjoint subsets and SNS is applied within each subset, wrapped in Gibbs sampling. This functionality is available via the `part` argument, which is a list containing the set of state space dimensions belonging to each subset. Convenience functions `sns.make.part` and `sns.check.part` allow users to easily create partition lists and check their validity, respectively.

3.3. Diagnostics

sns includes a rich set of diagnostics which can be accessed via functions `summary.sns` and `plot.sns`. Some of these are generic measures applicable to all MCMC chains, some are specific to MH-based MCMC algorithms, and some are even further specialized for SNS as a particular flavor of MH. Where possible, we have used the library **coda**, but opted to create and maintain an independent set of functions due to their specialized and extended nature.

MCMC diagnostics: In `summary.sns`, we calculate the usual MCMC chain summaries including mean, standard deviation, quantiles, and effective sample size. We also calculate a sample-based p-value for each coordinate. In `plot.sns` we have log-density trace plot, state vector trace plots, effective sample size by coordinate, state vector histograms, and state vector autocorrelation plots.

MH diagnostics: In `summary.sns`, we calculate the acceptance rate of MH transition proposals. If `mh.diag` flag is set to `TRUE`, all 4 components of the MH test (`log.p`, `log.p.prop`, `log.q` and `log.q.prop`) are returned as well.

SNS diagnostics: In `summary.sns`, we return `reldev.mean` (if `sns.run` was called with `mh.diag` set to `TRUE`), defined as the average relative deviation of log-density change (with respect to PDF maximum) from quadratic approximation (also constructed at PDF maximum). The location of PDF maximum is extracted from the Gaussian fit in the last iteration under NR mode. The higher this value, the more likely it is for the SNS to exhibit bad mixing. This is illustrated in Section 4. For `reldev.mean` to be valid, the user must ensure that the value of the argument `nnr` supplied to `sns.run` is sufficiently high to ensure convergence at the end of NR phase.

3.4. Full Bayesian Prediction

The function `predict.sns` allows for full Bayesian prediction, using a sample-based representation of predictive posterior distribution. It accepts an arbitrary function of state vector as argument `fpred`, and applies the function across all samples of state vector, supplied in the first argument, which must be an output of `sns.run`. The core philosophy in full Bayesian prediction is to postpone summarization of samples until the last step. For example, rather than supplying the expected values of coefficients into a function, we supply the samples and take the expected value after applying the function. Following this proper approach is important for several reasons:

1. Mathematically, an arbitrary function is not commutable with the expected value operator. Therefore, applying expected value early produces incorrect results.

2. For a similar reason, confidence intervals cannot be propagated through arbitrary functions. Therefore, correct uncertainty measurement also requires a full Bayesian approach.

4. Using sns

In this section, we illustrate how **sns** can be used via several examples. First, we launch an R session and load the **sns** package as well **mvtnorm** (used for evaluating the multivariate Gaussia log-density in example 1):

```
R> library(sns)
R> library(mvtnorm)
```

4.1. Example 1: Multivariate Gaussian

Using **sns** to sample from a multivariate Gaussian is a contrived, but pedagogical, example. Since log-density for a multivariate Gaussian is quadratic, its second-order Taylor series expansion is not approximate but exact. In other words, the proposal function becomes location-independent, and equal to the sampled distribution. This means that 1) the MH test is always accepted, and 2) consecutive samples are completely independent, and hence the resulting chain is no longer Markovian. Of course, since we know how to sample from multivariate Gaussian proposal functions, we might as well directly sample from the multivariate Gaussian distribution. Hence, the pedagogical nature of this example. To utilize **sns**, we must first implement the log-density and its gradient and Hessian:

```
R> logdensity.mvg <- function(x, mu, isigsq) {
+    f <- dmvnorm(x = as.numeric(x)
+      , mean = mu, sigma = solve(isigsq), log = TRUE)
+    g <- - isigsq %*% (x - mu)
+    h <- -isigsq
+    return (list(f = f, g = g, h = h))
+  }
```

We now draw 500 samples from this log-desity, using pre-specified values for `mu` (mean vector) and `isigsq` (inverse of the covariance matrix, or precision matrix) in a 3-dimensional state space:

```
R> K <- 3
R> mu <- runif(K, min = -0.5, max = +0.5)
R> isigsq <- matrix(runif(K*K, min = 0.1, max = 0.2), ncol = K)
R> isigsq <- 0.5*(isigsq + t(isigsq))
```

```
R> diag(isigsq) <- rep(0.5, K)
R> x.init <- rep(0.0, K)
R> x.smp <- sns.run(x.init, logdensity.mvg, niter = 500
+     , mh.diag = TRUE, mu = mu, isigsq = isigsq)
```

Next, we use the `summary.sns` function to view some of the diagnostics:

```
R> summary(x.smp)

Stochastic Newton Sampler (SNS)
state space dimensionality:  3
total iterations:  500
        NR iterations:  10
        burn-in iterations:  250
        end iteration:  500
        thinning interval:  1
        sampling iterations (before thinning):  250
acceptance rate:  1
        mean relative deviation from quadratic approx: 3.93e-15 % (post-burnin)
sample statistics:
        (nominal sample size: 250)
        mean         sd        ess       2.5%        50%  97.5% p-val
1  -0.497340   1.505447 250.000000  -3.099683  -0.743241 2.5354 0.696
2   0.033345   1.670739 250.000000  -3.151353   0.028756 3.0644 1.000
3   0.044322   1.476031 250.000000  -2.568027   0.097083 2.7600 0.944
summary of ess:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    250     250     250     250     250     250
```

As expected, the acceptance rate is 100%, and there is no deviation from quadratic approximation, for SNS sampling of a multivariate Gaussian.

In real-world applications, the Gaussian proposal function is only an approximation for the sampled distribution (since log-density is not quadratic), creating the Markovian dependency and less-than-perfect acceptance rate. We study one such example next.

4.2. Example 2: Bayesian Poisson regression

Generalized Linear Models (GLMs) (Nelder and Baker 1972) are an important family of statistical models with applications such as risk analysis (Sobehart *et al.* 2000), public health (Sharabiani *et al.* 2011) and political science (Gelman and Hill 2007). GLMs can be extended to incorporate data sparseness and heterogeneity via the Hierarchical Bayesian framework (Rossi *et al.* 2005) or to account for repeated measurements and longitudinal data via Generalized Linear Mixed Model (McCulloch 2006). With properly-chosen link functions, many GLMs are known - or can be easily proven - to have globally-concave log-densities with negative-definite Hessian matrices (Gilks and Wild 1992; Mahani and Sharabiani 2015a). As such, GLMs are excellent candidates for SNS. Embedded in Bayesian frameworks, they continue to enjoy log-concavity assuming the same property holds for prior terms, according to the Bayes' rule and the invariance of concavity for adding functions.

In our second example, we illustrate how to apply **sns** to the log-likelihood of Poisson regression. As before, we start with constructing the log-density, using the expander framework of **RegressionFactory** package:

```
R> library(RegressionFactory)
R> loglike.poisson <- function(beta, X, y) {
+    regfac.expand.1par(beta, X = X, y = y
+       , fbase1 = fbase1.poisson.log)
+ }
```

Now we simulate data from the generative model:

```
R> K <- 5
R> N <- 1000
R> X <- matrix(runif(N * K, -0.5, +0.5), ncol = K)
R> beta <- runif(K, -0.5, +0.5)
R> y <- rpois(N, exp(X %*% beta))
```

For reference, we do a maximum-likelihood (ML) estimation of the coefficients using `glm` command:

```
R> beta.init <- rep(0.0, K)
R> beta.glm <- glm(y ~ X - 1, family = "poisson", start = beta.init)$coefficients
```

As mentioned before, **sns** can be run in non-stochastic mode, which is equivalent to Newton-Raphson optimization with line search. Results should be identical, or very close, to `glm` results:

```
R> beta.sns <- sns.run(beta.init, fghEval = loglike.poisson
+    , niter = 20, nnr = 20, X = X, y = y)
R> beta.nr <- beta.sns[20, ]
R> cbind(beta.glm, beta.nr)
```

```
     beta.glm    beta.nr
X1 -0.4663463 -0.4663463
X2 -0.2170187 -0.2170187
X3 -0.5840998 -0.5840998
X4 -0.1261674 -0.1261674
X5  0.3182123  0.3182123
```

The primary use-case for **sns** is not ML estimation, but rather MCMC sampling of the distribution. To do this, we perform the first few iterations in non-stochastic mode (20 iterations here), and then switch to stochastic mode for the remaining 180 iterations:

```
R> beta.smp <- sns.run(beta.init, loglike.poisson
+    , niter = 200, nnr = 20, mh.diag = TRUE, X = X, y = y)
```

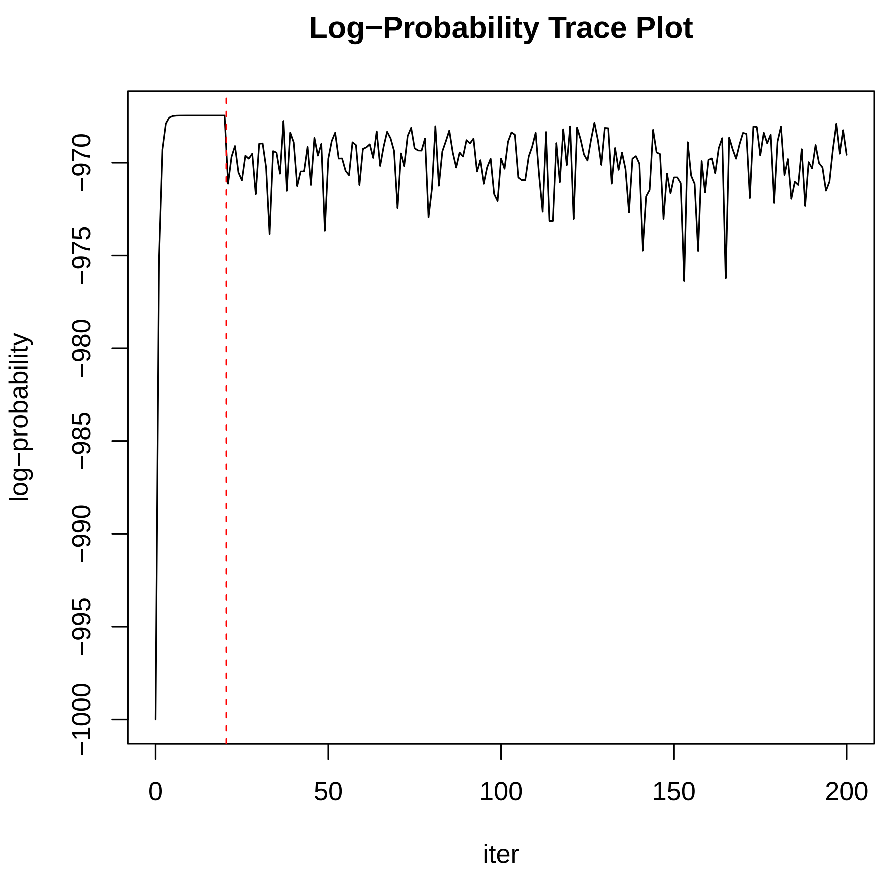

Figure 1: Log-probability trace plot for Poisson regression problem with $K = 5$ and $N = 1000$. Vertical line separates non-stochastic mode (left, 20 iterations) from stochastic mode (right, 180 iterations).

Examining the log-probability trace plot (Figure 1) shows an expected pattern: During non-stochastic phase (left of vertical line) log-probability rises steadily while approaching the peak. During MCMC sampling, on the other hand, PDF maximum forms an upper bound for the MCMC movements, and the chain occasionally visits low-probability areas. The plot is created using the following line:

```
R> plot(beta.smp, select = 1)
```

The `plot.sns` function offers 4 other types of plots, besides the log-probability trace plot. We refer the reader to the package documentation for details. Further diagnostic information can be accessed via the `summary.sns` function:

```
R> summary(beta.smp)
```

```
Stochastic Newton Sampler (SNS)
state space dimensionality:  5
```

```
total iterations:  200
        NR iterations:  20
        burn-in iterations:  100
        end iteration:  200
        thinning interval:  1
        sampling iterations (before thinning):  100
acceptance rate:  0.97
        mean relative deviation from quadratic approx: 0.33 % (post-burnin)
sample statistics:
        (nominal sample size: 100)
        mean         sd        ess       2.5%        50%   97.5%
1  -0.470714   0.111004 127.149650  -0.689517  -0.468404 -0.2765
2  -0.219657   0.117576 100.000000  -0.445595  -0.202877  0.0338
3  -0.562456   0.112047 142.837102  -0.793640  -0.564578 -0.3469
4  -0.106050   0.106618 100.000000  -0.293523  -0.105121  0.1310
5   0.305221   0.106901  64.291409   0.096068   0.311385  0.5082
  p-val
1  0.01 **
2  0.08 .
3  0.01 **
4  0.30
5  0.01 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
summary of ess:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  64.29  100.00  100.00  106.90  127.10  142.80
```

The `summary.sns` function discards the first half of the samples as burn-in by default, before calculating sample statistics and acceptance. This behavior can be controlled via the argument `nburnin`. Arguments `end` and `thin` have behavior similar to their counterparts in the `as.mcmc` function of **coda** package. We observe two numbers in the summary print-out: Firstly, acceptance rate is less than 100%, contrary to the case with a multivariate Gaussian PDF (example 1). Secondly, the mean relative deviation from quadratic approximation (`reldev.mean`) is now non-zero, again reflecting non-Gaussianity of the poisson likelihood PDF. The number (<1%) is still small, however, leading to high acceptance rate and good mixing of the chain.

Next, we want to predict the response variable in Poisson regression model, given new values of the explanatory variables. We distinguish between two types of prediction: 1) predicting mean response, 2) generating samples from posterior predictive distribution. We illustrate how to do both using the `predict.sns` function. We begin by implementing the mean prediction function, which simply applies the inverse link function (exponential here) to the linear predictor. (For better comparison between the two prediction modes, we increase number of samples to 1000)

```
R> beta.smp <- sns.run(beta.init, loglike.poisson
+    , niter = 1000, nnr = 20, mh.diag = TRUE, X = X, y = y)
```

```
R> predmean.poisson <- function(beta, Xnew) exp(Xnew %*% beta)
```

The following single line performs sample-based prediction of mean response (using `X` in lieu of `Xnew` for code brevity):

```
R> ymean.new <- predict(beta.smp, predmean.poisson
+     , nburnin = 100, Xnew = X)
```

`ynew` is a matrix of `N` (1000) rows and `niter - nburnin` (900) columns. Each row corresponds to an observation (one row of `Xnew`), and each column corresponds to a prediction sample (one row of `beta.smp` after burn-in).

We can also generate samples from posterior predictive distribution as follows:

```
R> predsmp.poisson <- function(beta, Xnew)
+     rpois(nrow(Xnew), exp(Xnew %*% beta))
R> ysmp.new <- predict(beta.smp, predsmp.poisson
+     , nburnin = 100, Xnew = X)
```

Comparing prediction summaries is illuminating:

```
R> summary(ymean.new)
R> summary(ysmp.new)

prediction sample statistics:
        (nominal sample size: 900)
         mean         sd        ess       2.5%        50%  97.5%
1    0.865031   0.064676 900.000000   0.742812   0.862279 1.0019
2    1.267430   0.106293 900.000000   1.077680   1.259413 1.4882
3    1.405566   0.073465 731.714663   1.261902   1.403256 1.5501
4    0.889331   0.063354 900.000000   0.773876   0.887845 1.0236
5    1.154694   0.081166 900.000000   1.009981   1.150145 1.3199
6    1.206132   0.076625 808.257228   1.071271   1.202108 1.3671
...

prediction sample statistics:
        (nominal sample size: 900)
        mean        sd       ess      2.5%       50% 97.5%
1    0.87333   0.93240 900.00000   0.00000   1.00000     3
2    1.28000   1.15218 900.00000   0.00000   1.00000     4
3    1.37444   1.19640 900.00000   0.00000   1.00000     4
4    0.87556   0.93864 900.00000   0.00000   1.00000     3
5    1.15000   1.02296 526.62253   0.00000   1.00000     3
6    1.23000   1.10987 900.00000   0.00000   1.00000     4
...
```

In the limit of infinite samples, the mean predictions from the two methods will be equal, and they are quite close based on 900 samples above. However, standard deviation of predictions is much larger for `predsmp.poisson` compared to `predmean.poisson`, as the former combines the uncertainty of coefficient values (represented in the `sd` values for `beta`'s) with the uncertainty of samples from Poisson distribution around the mean, i.e. the `sd` of Poisson distribution. Also note that, as expected, quantiles for `predsmp.poisson` are discrete since the predictions are discrete, while the quantiles for `predmean.poisson` are continuous as the predictions are continuous in this case.

4.3. Example 3: High-dimensional Bayesian Poisson regression

Contrary to standard Metropolis variants with multivariate Gaussians centered on current point, SNS is an aggressive, non-local MCMC algorithm as it seeks to construct a global, Gaussian approximation of the PDF. Under favorable conditions, this can lead to uncorrelated chains and efficient sampling, with the extreme case of perfectly uncorrelated samples for a multivariate Gaussian distribution. An important pathological case arises when the state space dimensionality is high. Continuing with the Poisson regression example, we increase K from 5 to 100, while holding $N = 1000$. To illustrate that the problem is not covergence but mixing, we explicitly use the `glm` estimate (mode of PDF) as the initial value for the MCMC chain:

```
R> K <- 100
R> X <- matrix(runif(N * K, -0.5, +0.5), ncol = K)
R> beta <- runif(K, -0.5, +0.5)
R> y <- rpois(N, exp(X %*% beta))
R> beta.init <- glm(y ~ X - 1, family = "poisson")$coefficients
R> beta.smp <- sns.run(beta.init, loglike.poisson
+    , niter = 100, nnr = 10, mh.diag = TRUE, X = X, y = y)
R> summary(beta.smp)
```

```
Stochastic Newton Sampler (SNS)
state space dimensionality:  100
total iterations:  100
        NR iterations:  10
        burn-in iterations:  50
        end iteration:  100
        thinning interval:  1
        sampling iterations (before thinning):  50
acceptance rate:  0.16
        mean relative deviation from quadratic approx: 8.35 % (post-burnin)
sample statistics:
        (nominal sample size: 50)
       mean        sd       ess      2.5%       50%   97.5% p-val
1  0.421594  0.109166  9.760501  0.164051  0.447386  0.5166  0.02 *
2  0.219483  0.105112  4.913961 -0.081762  0.223558  0.4021  0.12
3  0.203045  0.078498  5.749257  0.060661  0.250520  0.3404  0.02 *
4 -0.100247  0.069272  2.173084 -0.249282 -0.066971 -0.0368  0.04 *
5  0.111424  0.064391  4.121670  0.032764  0.122196  0.2168  0.02 *
6  0.115763  0.106802  5.238306 -0.064935  0.089167  0.2260  0.24
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
...
summary of ess:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.096   4.095   5.335   6.991   6.694  72.500
```

We see a significant drop in acceptance rate as well as effective sample sizes for the coefficients. Also note that mean relative deviation from quadratic approximation is now nearly 10x larger than the value for $K = 5$. To improve mixing, we use the 'state space partitioning' strategy of **sns**, available via the `part` argument of `sns` and `sns.run`. This leads to SNS sampling of subsets of state space wrapped in Gibbs cycles, with each subset being potentially much lower-dimensional than the original, full space. This strategy can significantly improve mixing:

```
R> beta.smp.part <- sns.run(beta.init, loglike.poisson
+    , niter = 100, nnr = 10, mh.diag = TRUE
+    , part = sns.make.part(K, 10), X = X, y = y)
R> summary(beta.smp.part)

Stochastic Newton Sampler (SNS)
state space dimensionality:  100
state space partitioning:  10  subsets
total iterations:  100
        NR iterations:  10
        burn-in iterations:  50
        end iteration:  100
        thinning interval:  1
        sampling iterations (before thinning):  50
acceptance rate:  0.94
sample statistics:
        (nominal sample size: 50)
        mean         sd        ess       2.5%        50%  97.5% p-val
1  0.3969166  0.0885804 39.1691610  0.2100833  0.3964180 0.5402  0.02
2  0.1896767  0.1152469 50.0000000 -0.0248610  0.1865183 0.3670  0.08
3  0.1579658  0.0947398 28.9506128 -0.0122672  0.1549021 0.3189  0.08
4 -0.1147680  0.0980224 29.6244795 -0.2697117 -0.1198215 0.0670  0.32
5  0.1540473  0.1003371 50.0000000 -0.0239445  0.1473975 0.3207  0.12
6  0.1466464  0.1027434 32.5836802 -0.0095312  0.1411849 0.3683  0.08

1 *
2 .
3 .
4
5
6 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
...
```

```
summary of ess:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.77   28.89   50.00   41.59   50.00   93.70
```

Notice the improved acceptance rate as well as effective sample sizes. A comparison of log-probability trace plots confirms better mixing after convergence to PDF mode (see Figure 2).

```
R> par(mfrow = c(1,2))
R> plot(beta.smp, select = 1)
R> plot(beta.smp.part, select = 1)
```

5. Summary

In this paper we presented **sns**, an R package for Stochastic Newton Sampling of twice-differentiable, log-concave PDFs, where a multivariate Gaussian resulting from second-order Taylor series expansion of log-density is used as proposal function in a Metropolis-Hastings framework. Using an initial non-stochastic mode, equivalent to Newton-Raphson optimization with line search, allows the chain to rapidly converge to high-density areas, while 'state space partitioning', Gibbs sampling of full state space in lower-dimensional blocks, allows SNS to overcome mixing problems while sampling from high-dimensional PDFs. There are several opportunities for further research and development.

Beyond twice-differentiability and log-concavity: Current version of SNS requires the log-density to be twice-differentiable and concave. In many real-world application, the posterior PDF does not have a negative-definite Hessian, or it cannot be proven to have such a property. In such cases, SNS would not be applicable to the entire PDF. However, if one can identify blocks within the full Hessian that does enjoy such property, then SNS can be combined with other, more generic sampling algorithms such as slice sampler or HMC, all embedded in a Gibbs cycle. The implementation would be similar to that of state space partitioning approach in **sns**, but replacing SNS with alternative samplers for some subsets. An alternative approach is discussed in Geweke and Tanizaki (2001) in the context of state-space models, where non-concave cases are handled by utilizing exponential and uniform distributions. Convergence and mixing properties of such extensions to general cases must be carefully studied. An important situation where twice-differentiability is violated is in the presence of boundary conditions. The current SNS algorithm assumes unconstrained state space. One way to deal with constrained subspaces is, again, to mix and match SNS and other samplers within Gibbs framework. For example, the slice sampler algorithm implemented in **MfUSampler** is capable of dealing with boxed constraints. It can therefore be assigned to subspaces with constraints, and the unconstrained subspaces can be handled by SNS. Further research is needed in order to relax twice-differentiability and log-concavity requirements for SNS.

Optimized state space partitioning:

Performance benchmarking:

There are several opportunities for further research and development:

1. Mixing properties of SNS must be better studied. Preliminary, one-dimensional analysis has provided sufficient conditions for the chain not to diverge too much from the

distribution mode, but extension to the real-world, high-dimensional case remains to be done. This understanding can in turn be used to better select the size of state space blocks in the partitioning strategy.

2. Currently, SNS requires strict global log-concavity for the PDF. Extending the algorithm to more general cases can improve its usefulness.

3. Currently, SNS cannot handle boundary conditions. Instead, they must be implemented via variable transformations, such as link functions in GLM models.

4. Impact of data centering and scaling on convergence and mixing of chains, not just for SNS but for MCMC algorithms in general, including Gibbs sampling.

References

Gamerman D (1997). "Sampling from the Posterior Distribution in Generalized Linear Mixed Models." *Statistics and Computing*, **7**(1), 57–68.

Gelman A, Hill J (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models.* Cambridge University Press.

Geman S, Geman D (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6), 721–741.

Geweke J, Tanizaki H (2001). "Bayesian estimation of state-space models using the Metropolis–Hastings algorithm within Gibbs sampling." *Computational Statistics & Data Analysis*, **37**(2), 151–170.

Gilks WR, Wild P (1992). "Adaptive Rejection Sampling for Gibbs Sampling." *Applied Statistics*, pp. 337–348.

Girolami M, Calderhead B (2011). "Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods." *Journal of the Royal Statistical Society B (Statistical Methodology)*, **73**(2), 123–214.

Hastings WK (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications." *Biometrika*, **57**(1), 97–109.

Mahani AS, Sharabiani MT (2015a). "Expander Framework for Generating High-Dimensional GLM Gradient and Hessian from Low-Dimensional Base Distributions: R Package RegressionFactory." *arXiv preprint arXiv:1501.06111.*

Mahani AS, Sharabiani MT (2015b). *MfUSampler: Multivariate-from-Univariate (MfU) MCMC Sampler.* R package version 0.9.2.

MATLAB (2014). *Release 2014b.* The MathWorks Inc., Natick, Massachusetts.

McCulloch CE (2006). *Generalized Linear Mixed Models.* Wiley Online Library.

Neal RM (2003). "Slice Sampling." *Annals of Statistics*, pp. 705–741.

Nelder JA, Baker R (1972). *Generalized Linear Models*. Wiley Online Library.

Nocedal J, Wright SJ (2006a). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer Verlag.

Nocedal J, Wright SJ (2006b). *Numerical Optimization*, chapter 3. Springer Series in Operations Research and Financial Engineering. Springer Verlag.

Plummer M (2013). *JAGS: Just Another Gibbs Sampler, Version 3.4.0*. URL `http://mcmc-jags.sourceforge.net/`.

Qi Y, Minka TP (2002). "Hessian-Based Markov Chain Monte-Carlo Algorithms." *1st Cape Cod Workshop on Monte Carlo Methods*.

Roberts GO, Rosenthal JS (1999). "Convergence of Slice Sampler Markov chains." *Journal of the Royal Statistical Society B (Statistical Methodology)*, **61**(3), 643–660.

Rossi PE, *et al.* (2005). *Bayesian Statistics and Marketing*. J. Wiley & Sons.

Sharabiani MTA, *et al.* (2011). "Immunologic profile of excessive body weight." *Biomarkers*, **16**(3), 243–251.

Sobehart JR, *et al.* (2000). "Moody's Public Firm Risk Model: A Hybrid Approach to Modeling Short-term Default Risk." *Moody's Investors Service, Global Credit Research, Rating Methodology, March*.

Thomas A, O'Hara B, Ligges U, Sturtz S (2006). "Making BUGS Open." *R News*, **6**(1), 12–17.

Affiliation:

Alireza S. Mahani
Scientific Computing Group
Sentrana Inc.
1725 I St NW
Washington, DC 20006
E-mail: `alireza.mahani@sentrana.com`

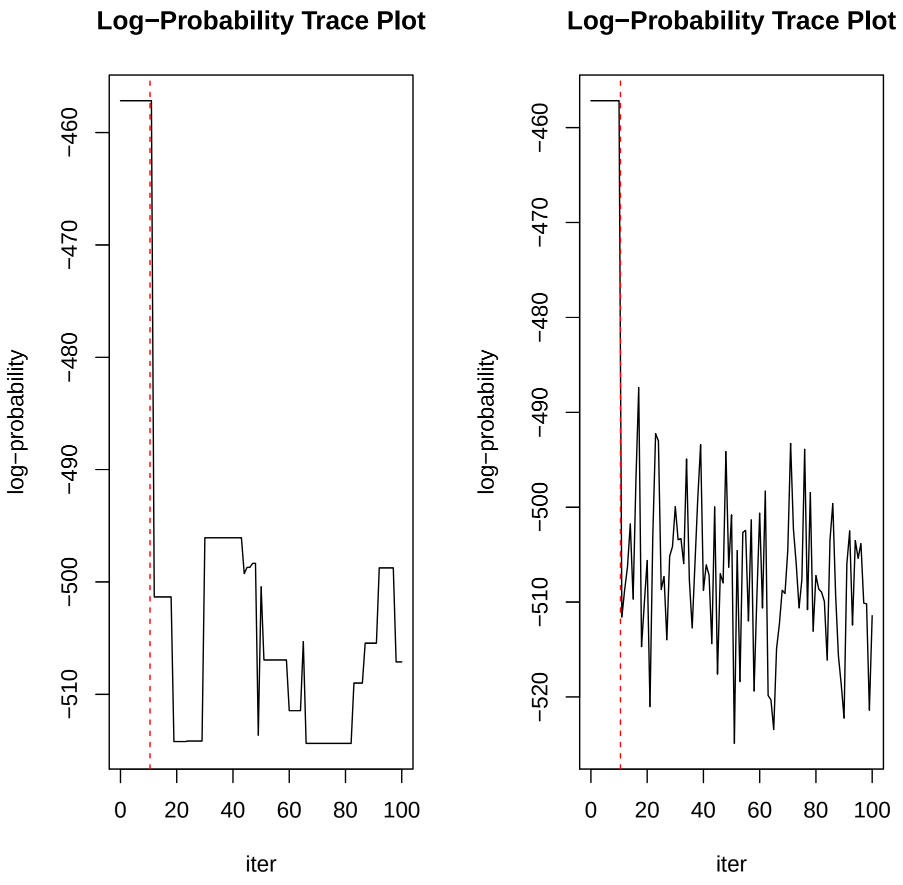

Figure 2: Comparison of log-probability trace plots for $N = 1000$ and $K = 100$, without (left) and with (right) state space partitioning using 10 subsets.