Marginal and Conditional algorithms for parameter inference in the α-stable distribution Matlab version

Marina Riabiz

March 2015

Contents

1 MAIN FUNCTIONS

Each of the following functions correspond to a scheme for posterior inference of the α and β parameters of the α-stable distribution. The first two functions implement the conditional Gibbs sampler scheme, in two different parametrization for the latent variables. The third and fourth routine run inexact marginal MCMC schemes, based on a numerical approximation of the likelihood, while the last two execute pseudo-marginal schemes, based on importance sampling. In details the proposal distribution for the latent variables is a piecewise constant envelope adaptively built.

1.1 MATLAB function: `[out, acc_a, acc_b] = conditional_v(z, N, Nc, a0, b0, pa, pb, sa, sb, MHsteps, plot)`

Given the standard stable data set `z`, of size `N`, this function performs the Metropolis within Gibbs sampler scheme introduced by [1] in the transformed parametrization for the latent variables $v = t_{\alpha\beta}(y)$. The chains $\{\alpha^{(k)}\}_{k=1}^{\mathrm{Nc}}$ and $\{\beta^{(k)}\}_{k=1}^{\mathrm{Nc}}$ have initial random values `a0` and `b0` respectively. In the current version the prior, proposal and posterior distributions are assumed to be bounded to $S_\alpha \in \{(0.1, 0.9), (1.1, 2)\}$ and $S_\beta \in \{(-1, 0), (0, 1)\}$, where S_α and S_β are selected in order to contain the initial values. Relaxation of this constraints is left to future analysis.

A variation of the MCMC scheme proposed by Buckle consists in the choice of the proposal distributions for α and β, which are not built by adaptive rejection sampling. We use instead a tunable mixture between a (truncated) Gaussian random walk, with probabilities `pa` and `pb` and standard deviations `sa` and `sb` respectively, and a uniform independent proposal with probabilities 1-`pa` and 1-`pb`. To compensate this arbitrary design of the proposals, we run the internal Metropolis within Gibbs step for `MHsteps`, where a typical value for `MHsteps` is 10. The variable `plot` is boolean, equal to 1 if the user wants to display the trace-plot showing the progress of the chains, 0 otherwise.

The variable `out` is a `Nc`$\times$3 cell array, containing $\{\mathbf{y}^{(k)}, \alpha^{(k)}, \beta^{(k)}\}_{k=1}^{\mathrm{Nc}}$, i.e. the set of latent variables in the original parametrization, and the values of the chains for α and β. Observe that the latent variables are initially sampled in the original parametrization through the subroutine `sampling_y`, then transformed to $\{v_i\}_{i=1}^{\mathrm{N}}$ and the Jacobian of the transformation is evaluated in the inverted variables (with a call of `inversion_step_ordered`) each time a new value of α or β is proposed. Finally `acc_a` and `acc_b` are `Nc`$\times$1 boolean vectors, whose components are 1 if the proposed values of the parameters have been accepted, 0 otherwise.

1.2 MATLAB function: `[out, acc_a, acc_b] = conditional_y(z, N, Nc, a0, b0, pa, pb, sa, sb, MHsteps, plot)`

This function is the counterparty of the previous one, but operates with latent variables in the original parametrization $\{y_i\}_{i=1}^{N}$. The target full conditional distributions of the parameters are spiked, which causes lack of convergence for the Markov chains.

1.3 MATLAB function: `[out, acc_a, acc_b] = GS_approx_marg(z, N, Nc, a0, b0, pa, pb, sa, sb, plot)`

This function implements an approximated marginal Metropolis within Gibbs sampler scheme. In details the likelihood is approximated using the `stablepdf` routine available in the `STABLE 5.3` Matlab library, that can be purchased from J.P. Nolan's academic webpage: http://academic2.american.edu/ jpnolan/stable/stable.html. See also [2] and [3] for a reference. It takes in input a standard stable dataset `z`, of lenght `N`, and it runs the chains for α and β, starting from the initial values `a0` and `b0`, for a total of `Nc` iterations. The proposal distributions are a tunable mixture between a (truncated) Gaussian random walk, with probabilities `pa` and `pb` and standard deviations `sa` and `sb` respectively, and a uniform independent proposal with probabilities 1-`pa` and 1-`pb`. The variable `plot` is boolean, equal to 1 if the user wants to display the trace-plot showing the progress of the chains, 0 otherwise. The third sub-plot represents the approximated log-likelihood.

The variable `out` is a `Nc`$\times$4 matrix, containing $\{\alpha^{(k)}, \beta^{(k)}, \tilde{z}^{(k)}, \tilde{\tilde{z}}^{(k)}\}_{k=1}^{\mathrm{Nc}}$, where $\tilde{z}^{(k)}$ is the approximated value of the log-likelihood in the current state of the chain $(\alpha^{(k)}, \beta^{(k)})$, while $\tilde{\tilde{z}}^{(k)}$ is the intermediate log-likelihood approximation after the Gibbs sampler step for α. Finally `acc_a` and `acc_b` are `Nc`$\times$1 boolean vectors, whose components are 1 if the proposed values of the parameters have been accepted, 0 otherwise.

1.4 MATLAB function: `[out, acc, cov_post] = MH_approx_marg(z, N, Ncs, Nb, Nc, a0, b0, p, sigma, plot)`

This function corresponds to the previous one, where a Metropolis Hastings scheme is used to jointly sample α and β. To capture the posterior covariance structure between the two parameters, a short run of the chain (`Ncs` iterations) is performed and the posterior covariance matrix is saved in `cov_post` after a burn-in of `Nb` iterations. Both the short and the long run (`Nc` iterations) of the chain have initial values `a0` and `b0`. The proposal distribution is a tunable mixture between a (truncated) Gaussian random walk, with probability `p` and a uniform independent proposal with probabilities 1-`p`. In the short run, in the random walk moves the parameters are drawn independently from truncated Gaussians with standard deviation `sigma`, while in the long run a bivariate truncated Gaussian with the estimated `cov_post` matrix is used.

The variable `out` is a `Nc`×3 matrix, containing $\{\alpha^{(k)}, \beta^{(k)}, \tilde{z}^{(k)}\}_{k=1}^{\mathrm{Nc}}$, where $\tilde{z}^{(k)}$ is the approximated value of the log-likelihood in the current state of the chain $(\alpha^{(k)}, \beta^{(k)})$. Finally `acc` is a `Nc`×1 boolean vector, whose components are 1 if the proposed values of the parameters have been accepted, 0 otherwise.

1.5 MATLAB function: `[out, acc_a, acc_b] = GS_pm(z, N, Nc, a0, b0, pa, pb, sa, sb, gl1, gl2, plot)`

This function performs the pseudo-marginal Gibbs sampler scheme. The parameters `out, acc_a, acc_b`, and `z, N, Nc, a0, b0, pa, pb, sa, sb, plot` have analogous meaning to those of the function `GS_approx_marg`. The likelihood is now not approximated numerically, but estimated via importance sampling, where `gl2` latent variables are drawn from a piecewise constant envelope built on $\pi(y|\alpha, \beta, z)$. For a fixed stable data point, each envelope has `gl1`+1 constant traits.

1.6 MATLAB function: `[out, acc, cov_post] = MH_pm(z, N, Ncs, Nb, Nc, a0, b0, p, sigma, plot)`

This function performs the pseudo-marginal Metropolis Hastings scheme. The parameters `out, acc, cov_post`, and `z, N, Ncs, Nb, Nc, a0, b0, p, sigma, plot` have analogous meaning to those of the function `MH_approx_marg`. The likelihood is now not approximated numerically, but estimated via importance sampling, where `gl2` latent variables are drawn from a piecewise constant envelope built on $\pi(y|\alpha, \beta, z)$. For a fixed stable data point, each envelope has `gl1`+1 constant traits.

1.7 Example

See `example.mat`. The standard stable dataset (100 samples) has $\alpha = 1.2$, $\beta = 0.8$. The initial values of the chains are $\alpha_0 = 1.7$, $\beta_0 = 0.4$, and the runs have 1000 iterations. When a Metropolis scheme is performed, the short run to determine the covariance between α and β has 1000 iterations, 500 of which are burn-in. Metropolis within Gibbs steps are executed 10 times. The Gaussian random walk components in the proposals for the parameters have probability 0.85 and standard deviations $\sqrt{10^{-3}}$; with probability 0.25 the values are proposed independently from a uniform distribution. 50 latent variables are drawn from piecewise constant envelopes built with 20 internal grid points for the importance sampler used in the pseudo-marginal schemes.

The user is asked to enter a number in $\{1, 2, \ldots, 6\}$, corresponding to each of the MCMC schemes presented above respectively.

2 AUXILIARY FUNCTIONS

The following functions are auxiliary to the previous ones. The first routine computes the sample posterior autocorrelation function, to be used as a diagnostic tool, while the second one is used to generate α-stable random variables. The third function outputs a set of latent variables in the original parametrization, given the transformed variables $\{v_i\}_{i=1}^N$, where each inversion is performed through `safe_newton`. The fourth function produces a noisy unbiased estimate of the α-stable likelihood via importance sampling, while the last one samples a set of latent variables $\{y_i\}_{i=1}^N$ from the respective full conditional, via adaptive rejection sampling.

2.1 MATLAB function: `sa = acf(y, lag, col)`

This is a version of the File Exchange - Matlab Central function by Calvin Price, which can be found at http://uk.mathworks.com/matlabcentral/fileexchange/30540-autocorrelation-function–acf-.
It computes `sa`, the sample autocorrelation function of the sequence `y`, through `lag` number of lags. It also produces a bar graph of the autocorrelation, with rejection region bands for testing if the individual autocorrelation is equal to 0.

2.2 MATLAB function: `x = CMS_weron(a, s, b, m, n)`

This function implements the Chambers Mallows Stuck method [4] for generating α-stable random variables, in the $S^1_\alpha(\sigma_2, \beta_2, \mu)$ parametrization according to Weron [5]. The set of parameters is (`a`, `s`, `b`, `m`) $= (\alpha, \sigma, \beta, \mu)$, and `n` is the number of i.i.d random variables generated and stored in the output vector `x`. In details for $\alpha \in (0, 2]$ and $\beta \in [-1, 1]$, if Y is uniformly distributed on $\left(-\frac{1}{2}, \frac{1}{2}\right)$, and W is an independent exponential random variable with mean 1, then

- for $\alpha \neq 1$

$$Z = t_{\alpha\beta_2}(Y)\, W^{\frac{\alpha-1}{\alpha}} \sim S_\alpha(1, \beta_2, 0) \tag{1}$$

- for $\alpha = 1$

$$Z = \left(\frac{\pi}{2} + \beta_2 \pi Y\right) \tan(\pi Y) - \beta_2 \log\left(\frac{W \cos(\pi Y)}{\frac{\pi}{2} + \beta_2 \pi Y}\right) \sim S_1(1, \beta_2, 0) \tag{2}$$

where

$$K(\alpha) = \alpha - 1 + \text{sign}(1 - \alpha) = \begin{cases} \alpha & \text{if } 0 < \alpha < 1 \\ \alpha - 2 & \text{if } 1 < \alpha < 2 \end{cases} \tag{3}$$

$$\eta_{\alpha\beta_2} = \beta_2 \frac{\pi}{2} K(\alpha) \tag{4}$$

$$t_{\alpha\beta_2}(y) = \frac{\sin(\pi\alpha y + \eta_{\alpha\beta_2})}{(\cos(\pi y))^{1/\alpha}} \left(\cos((\alpha - 1)\pi y + \eta_{\alpha\beta_2})\right)^{\frac{1-\alpha}{\alpha}}. \tag{5}$$

The variables obtained in this way have standard stable distribution ($\sigma_2 = 1$, $\mu = 0$). Multiplying Z with a non-negative constant σ_2 and adding a constant μ non standard stable random variable are obtained: $X = \sigma Z + \mu \sim S^1_\alpha(\sigma_2, \beta_2, \mu)$.

2.3 MATLAB function: `y = inversion_step_ordered(a, b, vI, I, n)`

Given `(a, b)`$=(\alpha, \beta)$ and given the relation $v_i = t_{\alpha\beta}(y_i)$, $i = 1, \ldots, n$, that associates the latent variables in their original parametrization $\{y_i\}_i$ with their transformed $\{v_i\}_i$, this function inverts a set of given variables `vI` $= \{v_i\}_i$ to `y` $= \{y_i\}_i$. In particular the `safe_newton` routine is called and to speed up the building of the interval bracketing the solution, `vI` are assumed to be pre-ordered in an ascending order (with the index `I` containing the mapping to the initial order, e.g. `[vI,I] = sort(v)`).

2.4 MATLAB function: `NL = noisy_likelihood(a, b, z, N, gl1, gl2)`

Returns `NL`, the value of the (not unbiased) log-likelihood corresponding to the noisy unbiased estimate of the likelihood referred to the parameters `(a, b)` $= (\alpha, \beta)$ and the standard stable data `z`. The likelihood value is produced by importance sampling, where each proposal distribution for the latent variables $\{y_i\}_{i=1}^{\text{gl2}}$ is a piecewise constant envelope on the conditional (unnormalized) distribution $\pi(y|\alpha, \beta, z)$. Each envelope has `gl1`+1 levels.

2.5 MATLAB function: `[x,xEvals,test,nEvals] = safe_newton(fun,der,a,b,tx,tf,nMax)`

Computes the zero of the one-variable scalar function `fun` (with derivative `der`) based on the Newton method. An initial interval `(a, b)` bracketing the solution is required. This is tested with a sign condition suitable for monotonic functions, as happens for $t_{\alpha,\beta}$; if this is not the case, an error message is displayed, the variable `test` is assigned a value of 1, and the execution of the function is interrupted. To guarantee that each of the found intervals brackets the zero, each time the Newton step does not bracket the solution, a bisection step is taken instead. The procedure is repeated until the amplitude of the bracketing interval is bigger than the tolerance `tx`, or the absolute value of the function in the estimated solution in higher than the tolerance `tf`, or the number of iterations has not reached the allowed maximum `nMax`. Approximations of the root of `fun` are saved in `xevals`, while `nEvals` contains the effective number of iterations performed.

2.6 MATLAB function: `y = sampling_y(a,b,z,n)`

This function samples `n` latent variables $\{y_i\}_{i=1}^{n}$, as a step of the Gibbs sampler scheme described in [1]. One latent variable is drawn by adaptive rejection sampling from the full conditional $\pi(y|\alpha, \beta, z_i)$, for each standard stable data point z_i. In details $\pi(y|\alpha, \beta, z_i)$ is unimodal and it is possible to build a piecewise constant envelope with the rejected points, until the first acceptance happens. The value of the parameters (α, β) are contained in `a` and `b`.

References

[1] D. J. Buckle, "Bayesian inference for Stable distributions," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 605–613, 1995.

[2] J. P. Nolan, "Numerical calculation of Stable densities and distribution functions," *Comm. Statist. Stochastic Models*, vol. 13, no. 4, pp. 759–774, 1997.

[3] J. P. Nolan, *Stable Distributions - Models for Heavy Tailed Data.* Boston: Birkhauser, 2015, in progress, Chapter 1 online at academic2.american.edu/∼jpnolan.

[4] J. M. Chambers, C. L. Mallows, and B. W. Stuck, "A method for simulating Stable random variables," *Journal of the American Statistical Association*, vol. 71, no. 354, pp. 340–344, 1976.

[5] R. Weron, "On the Chambers-Mallows-Stuck method for simulating skewed Stable random variables," *Statistics & Probability Letters*, vol. 28, no. 2, pp. 165 – 171, 1996.