MCMC Computations of Horseshoe Normal Mean Model

March 2024

Ziang Fu

1 Introduction

In the field of Bayesian High Dimensional Statistics, **horseshoe** prior plays an important role in variable selection, high dimensional statistics sparsity, change point detection, etc. Horseshoe is an elegant model, but its computation remains a troublesome problem even after 2010 [1]. Here below is a note on studying Bayesian computation with the example of the horseshoe normal mean model, a classical and simple model discussed in horseshoe problems.

2 Horseshoe Normal Mean Model

The Horseshoe Normal Mean Model[6] is a rather simple model, so it is easier to test the effects of different computation methods on horseshoe prior. Here we present the model below, assuming sample Y(a T dimensional vector) is conducted by signal β and normal white noise ϵ:

$$\begin{aligned}
\text{Model:}\quad & Y=\beta+\varepsilon \\
\text{s.t.}\quad & \varepsilon \sim N_T\left(0, \sigma^2 I_T\right) \\
& p(\sigma) \propto \sigma^{-1} \\
& \beta \sim N_T\left(0, \sigma^2 \tau^2 \Lambda^2\right) \\
& \tau \sim C^{+}(0,1) \\
& \lambda_t \sim C^{+}(0,1)
\end{aligned}$$

Where σ^2 refers to the variance of white noise, τ refers to the global shrinkage parameter, and $\Lambda=\operatorname{diag}\left(\lambda_1, \lambda_2, \cdots, \lambda_T\right)$ refers to the local shrinkage parameters (since this note is a preparation for later Bayesian Changepoint Detection Research, here we use t to indicate the t^{th} element of a vector). $C^{+}(0,1)$ refers to the standard half-Cauchy distribution.

3 Computation

Although the model seems quite simple, the computation is difficult, especially when sampling half-Cauchy distribution. Here we provide 5 ways of hybrid MCMC, which have been coded in **hs_nm.R**, then we will compare these methods with **HS.normal.means()** in **horseshoe** package[7].

3.1 Gibbs Steps for σ^2 And β

In all the 5 approaches, σ^2 And β are sampled with Gibbs steps, whose posterior distributions are provided below:

$$
\begin{aligned}
&\left[\sigma^2 \mid \cdot\right] \sim IG\left(T, \frac{1}{2}\sum_{t=1}^{T}\left(y_t-\beta_t\right)^2+\frac{1}{2}\sum_{t=1}^{T}\left(\frac{\beta_t}{\lambda_t \tau}\right)^2\right)\\
&[\beta \mid \cdot] \sim N_T\left(\left(I+\left(\tau^2\Lambda^2\right)^{-1}\right)^{-1}\left(\tau^2\Lambda^2\right)Y, \sigma^2\left(I+\left(\tau^2\Lambda^2\right)^{-1}\right)^{-1}\right)\\
&\left(\left[\beta_t \mid \cdot\right] \sim N\left(\frac{\tau^2\lambda_t^2 y_t}{1+\tau^2\lambda_t^2}, \frac{\tau^2\lambda_t^2\sigma^2}{1+\tau^2\lambda_t^2}\right)\right)
\end{aligned}
$$

Here IG refers to inverse-Gamma distribution, and we implement a blocked Gibbs sampler to sample β (same as **HS.normal.means()**).

3.2 Wand Mixture Sampling τ And Λ

Makalic and Schmidt (2015)[4] provided a simple sampler using Wand Mixture solving half-Cauchy sampling in horseshoe prior only through Gibbs steps. This simple data augmentation trick seems surprisingly cheerful, however, its computational performance may not be as fantastic as its formulas. Here we first briefly introduce Wand Mixture, then give its Gibbs samplers.

$$
\text{Wand Mixture:}\quad \mathrm{X} \sim \mathrm{C}^{+}(0,\tau) \Leftrightarrow \mathrm{X}^2 \mid \gamma \sim \mathrm{IG}(1/2, 1/\gamma), \gamma \sim \mathrm{IG}\left(1/2, 1/\tau^2\right)
$$

Thus, Gibbs samplers of τ and Λ(elementwise) can be easily induced:

$$
\begin{aligned}
&\left[\lambda_t^2 \mid \cdot\right] \sim IG\left(1, \frac{1}{v_t}+\frac{1}{2}\left(\frac{\beta_t}{\sigma\tau}\right)^2\right)\\
&[v \mid \cdot] \sim IG\left(1, 1+\frac{1}{\lambda_t^2}\right)\\
&\left[\tau^2 \mid \cdot\right] \sim \mathrm{IG}\left(\frac{T+1}{2}, \frac{1}{\eta}+\frac{1}{2}\sum_{t=1}^{T}\left(\frac{\beta_t}{\lambda_t\sigma}\right)^2\right)\\
&[\eta \mid \cdot] \sim IG\left(1, 1+\frac{1}{\tau^2}\right)
\end{aligned}
$$

This method can be selected in our function **hs_nm()** when setting parameter **method="gibbs"**.

3.3 Metropolis-within-Gibbs Sampler

Gilks, W. R. et al.(1995)[3] proposed a metropolis-within-Gibbs sampler to improve the performance of the Gibbs sampler by replacing some Gibbs steps with Metropolis-Hasting steps. Here we use random-walk Metropolis-Hasting steps to sample τ and Λ. We first implement inverse square transformation to τ and λ_t.

Define $\eta = 1/\tau^2$ and $\xi_t = 1/\lambda_t^2$, we have

$$[\eta \mid \cdot] \propto \frac{\eta^{\frac{T-1}{2}}}{1+\eta} \exp\left\{-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{\beta_t}{\sigma\lambda_t}\right)^2 \eta\right\} := f_1(\eta)$$
$$[\xi_t \mid \cdot] \propto \exp\left\{-\frac{\mu_t^2}{2}\xi_t\right\}\frac{1}{1+\xi_t} := f_2(\xi_t)$$

And the random-walk Metropolis-Hasting steps of τ and λ_t are as follows (in iteration k):

Algorithm 1: Random-Walk Metropolis-Hasting Steps

Data: $\eta^{(k-1)}$
Result: $\eta^{(k)}$
1. Generate a proposal value $\eta^\star$ with random-walk step: $\eta^\star = \eta^{(k-1)} + z$, where $z \sim N(0, 0.04)$;
2. Compute acceptance rate: $\alpha = \min\left\{1, \frac{f_1(\eta^\star)}{f_1(\eta^{(k-1)})}\right\}$;
3. $\eta^{(k)} = \begin{cases} \eta^\star & \text{if } \mathrm{Unif}(0,1) \le \alpha \\ \eta^{(k-1)} & \text{otherwise} \end{cases}$.

And ξ_t is sampled with similar steps. Here we should notice that η and λ_t are positive, so if the proposal value is occasionally less than 0 (which is nearly impossible), we use its absolute value.

This method can be selected in our function **hs_nm()** when setting parameter **method="MwG"**.

3.4 Slice Samplers Sampling τ And Λ

3.4.1 Damien's Slice Sampler

Here we implement two kinds of slice samplers to sample τ and Λ. One was published by Damien et al.(1999)[2], and the other was proposed by Neal, R.(2003)[5]. The first one seems to be a faster sampler, however, the other one performs better.
We first introduce Damien's slice sampler of λ_t. Since the posterior density of λ_t can be written as:

$$[\lambda_t \mid \cdot] \propto \frac{2}{\pi\left(1+\lambda_t^2\right)} \cdot \frac{1}{\lambda_t} \cdot \exp\left(-\frac{1}{\lambda_t^2}\left(\frac{\beta_t}{2\tau\sigma}\right)^2\right)$$

Define $\xi_t = 1/\lambda_t^2$ and $\mu_t = \beta_t/(\sigma\tau)$, we have:

$$[\xi_t \mid \cdot] \propto \exp\left\{-\frac{\mu_t^2}{2}\xi_t\right\}\frac{1}{1+\xi_t}$$

which can be seen as a combination of a non-increasing function $1/(1+\xi_t)$ and an exponential distribution with rate $\mu_t^2/2$. So we have the following algorithm:

Algorithm 2: Damien's Slice Sampler

Data: $\xi_t^{(k-1)}$
Result: $\xi_t^{(k)}$
1. Sample $u|\xi_t^{(k-1)} \sim Unif(0, \frac{1}{1+\xi_t^{(k-1)}})$;
2. Sample $\xi_t^{(k)}$ from $Exp(\frac{\mu_t^2}{2})$ truncated in the interval $(0, \frac{1}{u} - 1)$, which can be divided into the following steps;
2.1. Sample $v \sim Unif(0, 1 - exp(-\frac{\mu_t^2}{2}(\frac{1}{u} - 1)))$;
2.2. $\xi_t^{(k)} = -\log(1-v)/(\mu_t^2/2)$;

τ can be sampled with similar steps, however, the only difference is the distribution at step 2, which is no longer exponential, but a gamma distribution, $Gamma(\frac{(T+1)}{2}, \frac{\sum_{t=1}^{T}(\frac{\beta_t}{\lambda\sigma})^2}{2})$.

This method can be selected in our function **hs_nm()** when setting parameter **method="slice2"**.

3.4.2 Stepping-out Slice Sampler

The other slice sampler by Neal, R. transformed sampling a half-Cauchy distribution into two uniform distributions, here we introduce its steps with τ. Before slice sampling, we implement a logarithm transformation to τ.

Define $\eta = \log(\tau)$,

$$[\eta \mid \cdot] \propto \frac{e^{-(T-1)\eta}}{1+e^{2\eta}} \exp\left\{-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{\beta_t}{\lambda_t\sigma}\right)^2 e^{-2\eta}\right\} := f(\eta)$$

And the algorithm comes as follows (in iteration k):

Algorithm 3: Stepping-out Slice Sampler

Data: $\eta^{(k-1)}$
Result: $\eta^{(k)}$
1. Sample $u|\eta^{(k-1)} \sim Unif(u : f(\eta^{(k-1)} > u))$;
2. Sample $\eta^{(k)}|u \sim Unif(S), S := \{\eta : f(\eta) > u\}$, this step is hard to find the set S, the specific steps are as below;
2.1. Stepping-out: Searching for potential interval $I = [L, R]$ that contains S (setting w as window width(speed of extension) and m as the largest number of windows);
2.1.1. Initial Interval;
Sample $u_1 \sim Unif(0, 1)$;
$L = \eta^{(k-1)} - wu_1$;
$R = L + w$;
2.1.2. Limit of extension: Sample $v \sim Unif(0, 1)$;
$L = \eta^{(k-1)} - wu_1$;
$K = (m - 1) - J$;
while *($J > 0$ and $u_1 < f(L)$)* **do**
  (Left border) $L = L - w$;
  $J = J - 1$;
end
while *($K > 0$ and $u_1 < f(R)$)* **do**
  (Right border) $R = R + w$;
  $K = K - 1$;
end
2.2. Shrinkage: Given $\eta^{(k-1)}, u_1, L, R$, to sample $\eta^{(k)}$;
$\bar{L} = L, \bar{R} = R, Accept = 0$;
while *($!Accept$)* **do**
  Sample $u_2 \sim Unif(0, 1)$;
  $\eta^{(k)} = \bar{L} + u_2(\bar{R} - \bar{L})$;
  if *($u_1 < f(\eta^{(k)})$)* **then**
    $Accept = 1$;
  else if *($\eta^{(k)} < \eta^{(k-1)}$)* **then**
    $\bar{L} = \eta^{(k)}$;
  else
    $\bar{R} = \eta^{(k)}$;
  end
end

There are two difficulties in implementing this stepping-out slice sampling algorithm, one is how to choose w and m, and the other is the form of $f(\eta)$ (e.g. log-transformation λ_t is hard to converge).

This method can be selected in our function **hs_nm()** when setting parameter **method="slice"**

(slice sampler for λ_t is given by Damien's slice sampler).

3.5 Slice-Metropolis-within-Gibbs Sampler

In our function **hs_nm()**, the parameter setting **method="SMwG"** refers to the Slice-Metropolis-within-Gibbs Sampler, where τ is sampled by stepping-out slice sampler and λ_t is sampled with Metropolis-Hasting steps.

4 Simulation Results And Discussion

HS.normal.means() from **horseshoe** package implements **parameter expansion**[8] tricks for half-Cauchy sampling. Here we compare computation methods in **HS.normal.means()** and **hs_nm()** in computation stability, efficiency, and in a signal-noise classification problem.

Our data setting in this note is with one signal stage, where $signal = 2, 5, 10$, and two noise stages at two sides(more results in the folder **example 1**). And another example of multi-signal stages can be found in the folder **example 2**.

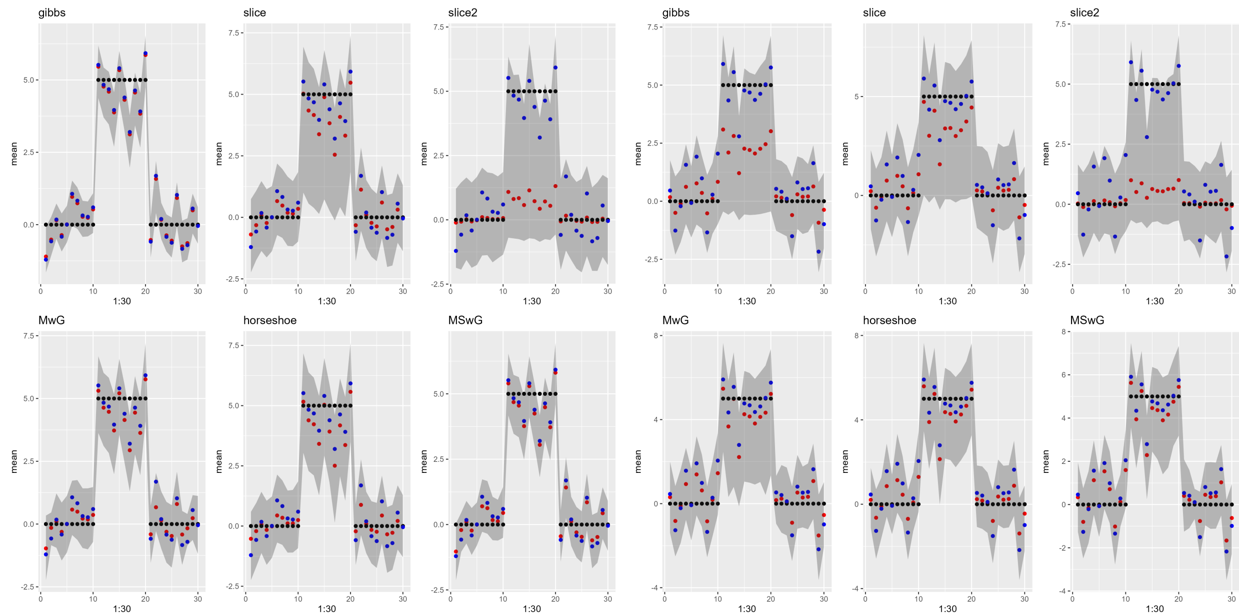
$$
\begin{aligned}
\text{data:} \quad & Y = \beta + \epsilon \\
\text{example 1:} \quad & \beta = \begin{cases} 0, & t = 1-10; 21-30 \\ signal, & t = 11-20 \end{cases} \\
\text{example 2:} \quad & \beta = \begin{cases} 0, & t = 1-10; 21-30; 41-50 \\ signal, & t = 11-20 \\ \frac{signal}{2}, & 31-40 \end{cases}
\end{aligned}
$$

4.1 Computation Stability And Efficiency

We can get the posterior distribution of parameters in the Horseshoe Normal Mean Model estimated by all 6 approaches. However, the computation stability is different. With different random seed settings, some estimated posterior distributions are stable(close), but some are not stable, which even affects the shrinkage effect and signal-noise classification. Some algorithms cannot classify signals and noises with any random seed, some can well classify with some random seeds, and some can with any random seed. This refers to different efficiency of algorithms.

We can see compared with **HS.normal.means()**, **"gibbs"**(Wand Mixture), is rather unstable, and the other algorithms related to slice sampler and Metropolis-Hasting are rather stable. **"slice2"**, **"slice"** are not as efficient as **"SMwG"** and **HS.normal.means()**.

σ^2 and τ are the two global parameters in the model, representing the efficiency of estimation and shrinkage effect. Here we use the mean in 200 random seeds of $var(\tau)$ to indicate computation



(a) Signal=5,Seed=2004

(b) Signal=5,Seed=2007

Figure 1: Computation Stability Example

stability, and the mean of $var(\sigma^2)$ indicates computation efficiency. The smaller the mean is, the more stable(efficient) the algorithm is. Here we can see from the tables below of computation stability and efficiency(signal=5) (others can be found in **compu S&E.txt**).

We can see from Table 1 and Table 2, in the aspect of computation stability, **"gibbs"** method is the most unstable one and far from others. **"slice2"** and **"MwG"** are less stable than **"slice"**, **HS.normal.mean()** and **"MSwG"**. **"MSwG"** and **"slice"**, representing stepping-out slice sampler, perform even better at computation stability and τ estimation. Besides, the computation efficiency of **"slice2"**, **"slice"**, **"gibbs"** performance pooler than the other 3 methods. Combining stepping-out slice sampler and random-walk Metropolis-Hasting sampler, **"MSwG"** performs almost as good as **HS.normal.means()** in computation efficiency, and even better in computation stability. This can be proved in Autocorrelation Function analysis, where **HS.normal.means** quickly drops below CI in $sigma^2$ ACF and stepping-out slice sampler (**"slice"** and **"MSwG"**) drops even much quicker in τ ACF.

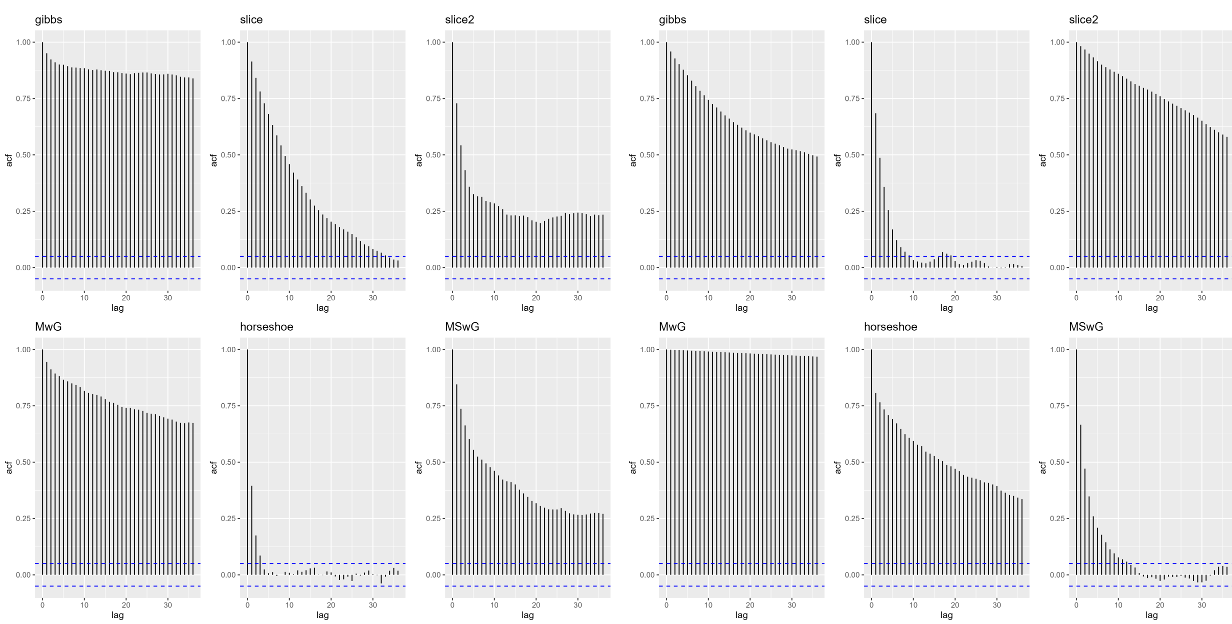
Computation stability can also be proved that although ACF of τ drops more slowly when signal becomes more significant, stepping-out slice sampler keeps a quick speed of dropping at a wide range of signal intensity. ACF of σ^2 shows that Metropolis-Hasting steps lowers MCMC convergence rate of **"MSwG"** compared with **"slice"**, **"MSwG"** still performs better especially in signal-noise classification problem.

method	mean	var
gibbs	1.3400140	5.90167892
slice	2.4150739	6.32607163
slice2	8.4280189	20.33815166
MwG	1.2136388	1.44721169
horseshoe	1.0118875	0.09931485
MSwG	0.6221907	0.16480728

Table 1: Sigma2 Posterior Distribution, Signal=5, Seed: 2004-2203

method	mean	var
gibbs	16.0260686	19682.29
slice	1.6658291	0.1438
slice2	0.5782571	9.2289
MwG	3.7004621	13.1300
horseshoe	1.5135995	0.2659
MSwG	1.7025372	0.1416

Table 2: Tau Posterior Distribution, Signal=5, Seed: 2004-2203



(a) sigma2,Signal=5,Seed=2009

(b) tau,Signal=5,Seed=2009

Figure 2: ACF of σ^2 and τ

4.2 Signal-Noise Classification

In order to further test the computation methods, we put them in a signal-noise classification problem, where a signal is defined if the true value μ is larger to smaller than 0, otherwise, it is a noise. And we classify it with CI(Credible Interval) criteria, i.e. if the multiple of lower bound and upper bound is larger than 0, then it is a signal, otherwise, it is a noise. We evaluate efficiency of each algorithm with precision, recall, and $F1$. Still taking 200 random seeds, we can see classification efficiency as below:

method	precision	recall	$F1$
gibbs	0.8928571	0.0200213561	0.0391644909
slice	0.8534704	0.0839443742	0.1528545120
slice2	1.0000000	0.0002716653	0.0005431831
MwG	0.8577406	0.1019393337	0.1822222222
horseshoe	0.9883721	0.0441787942	0.0845771144
MSwG	0.7124654	0.2891837194	0.4113883557

Table 3: Signal-Noise Classification, Signal=2, Seed: 2004-2203

method	precision	recall	F1
gibbs	0.7726948	0.226957800	0.350860110
slice	1.0000000	0.120879121	0.215686275
slice2	1.0000000	0.002244949	0.004479841
MwG	0.8253165	0.308420057	0.449035813
horseshoe	0.9913880	0.329461279	0.494566591
MSwG	0.9091324	0.343750000	0.498872463

Table 4: Signal-Noise Classification, Signal=5, Seed: 2004-2203

method	precision	recall	F1
gibbs	0.6387736	0.4107620	0.5000000
slice	1.0000000	0.3270525	0.4929006
slice2	0.9898305	0.1799877	0.3045897
MwG	0.7186312	0.3669903	0.4858612
horseshoe	0.9866798	0.3348401	0.5000000
MSwG	0.9376465	0.3408897	0.5000000

Table 5: Signal-Noise Classification, Signal=10, Seed: 2004-2203

Results in Table 3-5 show us that **"slice2"** performs poor in all 3 signal settings, **"gibbs"** may perform rather well in significant signals, however, due to its poor computation stability, its good performance only happens occasionally. Besides, although **"slice"** and **"MwG"** performs not well in slight signals, their combination **"MSwG"** performs really well, even much better than **HS.normal.means()** when $signal = 2$ in $F1$ scale. It also performs as well as **HS.normal.means()** when the signal becomes more significant, but it has a higher rate of FP, classifying noises as signals. stepping-out slice sampler of λ_t is hard to converge, however, if we can implement this sampler, computation efficiency and stability may even get further improved.

References

[1] C. M. Carvalho, N. G. Polson, and J. G. Scott. The horseshoe estimator for sparse signals. *Biometrika*, 97:465–480, 2010.

[2] P. Damien, J. Wakefield, and S. Walker. Gibbs sampling for bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(2):331–344, 1999.

[3] W. R. Gilks, N. G. Best, and K. K. C. Tan. Adaptive Rejection Metropolis Sampling Within Gibbs Sampling. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 44(4):455–472, 12 2018.

[4] E. Makalic and D. F. Schmidt. A simple sampler for the horseshoe estimator. *IEEE Signal Processing Letters*, 23(1):179–182, 2016.

[5] R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705 – 767, 2003.

[6] S. Pas, B. Kleijn, and A. Vaart. The horseshoe estimator: Posterior concentration around nearly black vectors. *Electronic Journal of Statistics*, 8, 04 2014.

[7] A. C. A. B. M. v. d. P. S van der Pas, J Scott. Implementation of the horseshoe prior. 0.2.0, 10 2022.

[8] J. G. Scott. Parameter expansion in local-shrinkage models. *arXiv*, 2010.