Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

A subsampling approach for Bayesian model selection

Jon Lachmann[a], Geir Storvik[b], Florian Frommlet[c], Aliaksandr Hubin[b,*]

[a] Department of Statistics, Stockholm University, Stockholm, Sweden
[b] Department of Mathematics, University of Oslo, Oslo, Norway
[c] CEMSIIS, Medical University of Vienna, Vienna, Austria

ARTICLE INFO

Article history:
Received 31 January 2022
Received in revised form 11 June 2022
Accepted 28 August 2022
Available online 13 September 2022

Keywords:
Bayesian model selection
Bayesian model averaging
Subsampling
MCMC
Tall data

ABSTRACT

It is common practice to use Laplace approximations to decrease the computational burden when computing the marginal likelihoods in Bayesian versions of generalised linear models (GLM). Marginal likelihoods combined with model priors are then used in different search algorithms to compute the posterior marginal probabilities of models and individual covariates. This allows performing Bayesian model selection and model averaging. For large sample sizes, even the Laplace approximation becomes computationally challenging because the optimisation routine involved needs to evaluate the likelihood on the full dataset in multiple iterations. As a consequence, the algorithm is not scalable for large datasets. To address this problem, we suggest using stochastic optimisation approaches, which only use a subsample of the data for each iteration. We combine stochastic optimisation with Markov chain Monte Carlo (MCMC) based methods for Bayesian model selection and provide some theoretical results on the convergence of the estimates for the resulting time-inhomogeneous MCMC. Finally, we report results from experiments illustrating the performance of the proposed algorithm.

1. Introduction

The Marginal Likelihood plays an extremely important role in Bayesian statistics. For a dataset $\boldsymbol{y}$ and a model $\mathfrak{m}$ which may depend on some unknown parameter vector $\boldsymbol{\theta}_\mathfrak{m}$, the marginal likelihood is given by

$$p(\boldsymbol{y}|\mathfrak{m}) = \int_{\boldsymbol{\theta}_\mathfrak{m}} p(\boldsymbol{y}|\mathfrak{m}, \boldsymbol{\theta}_\mathfrak{m}) p(\boldsymbol{\theta}_\mathfrak{m}|\mathfrak{m}) d\boldsymbol{\theta}_\mathfrak{m} \tag{1}$$

where $p(\boldsymbol{\theta}_\mathfrak{m}|\mathfrak{m})$ is the prior for $\boldsymbol{\theta}_\mathfrak{m}$ under model $\mathfrak{m}$ while $p(\boldsymbol{y}|\mathfrak{m}, \boldsymbol{\theta}_\mathfrak{m})$ is the likelihood function conditional on $\boldsymbol{\theta}_\mathfrak{m}$. Consider the comparison of two models $\mathfrak{m}_i$ and $\mathfrak{m}_j$ based on the ratio of their posterior marginal model probabilities:

$$\frac{p(\mathfrak{m}_i|\boldsymbol{y})}{p(\mathfrak{m}_j|\boldsymbol{y})} = \frac{p(\boldsymbol{y}|\mathfrak{m}_i)}{p(\boldsymbol{y}|\mathfrak{m}_j)} \times \frac{p(\mathfrak{m}_i)}{p(\mathfrak{m}_j)}. \tag{2}$$

* Corresponding author at: University of Oslo, 0851 Moltke Moes vei 35 Oslo, Norway.
E-mail addresses: jon.lachmann@stat.su.se (J. Lachmann), aliaksah@math.uio.no (A. Hubin).

https://doi.org/10.1016/j.ijar.2022.08.018

The ratio between the marginal likelihoods on the right-hand side is called the Bayes Factor [27]. It is commonly used to quantify to which extent one model is preferable compared to the other. To obtain the ratio (2) the marginal likelihoods need to be calculated up to a constant.

In the setting of Bayesian model averaging, if we are interested in some quantity Δ, we want to calculate the posterior marginal distribution

$$p(\Delta|\boldsymbol{y}) = \sum_{\mathfrak{m}\in\mathcal{M}} p(\Delta|\mathfrak{m}, \boldsymbol{y})p(\mathfrak{m}|\boldsymbol{y}). \tag{3}$$

To this end we need the posterior marginal model probability $p(\mathfrak{m}|\boldsymbol{y})$ for model $\mathfrak{m}$ which according to Bayes theorem can be calculated as:

$$p(\mathfrak{m}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\mathfrak{m})p(\mathfrak{m})}{\sum_{\mathfrak{m}'\in\mathcal{M}} p(\boldsymbol{y}|\mathfrak{m}')p(\mathfrak{m}')}. \tag{4}$$

Again, we need to obtain the marginal likelihood $p(\boldsymbol{y}|\mathfrak{m})$. Further, in the denominator, we would need to compute the marginal likelihood for every model $\mathfrak{m}' \in \mathcal{M}$. This is usually not computationally feasible except for very limited model spaces. A common solution to this is to use a Metropolis-Hastings algorithm to search through the model space within a Monte Carlo setting [20]. The required acceptance ratio when considering a move to $\mathfrak{m}^\star$ based on a proposal distribution $q(\mathfrak{m}^\star|\mathfrak{m})$ is then of the form

$$r_m(\mathfrak{m}, \mathfrak{m}^\star) = \min\left\{1, \frac{p(\boldsymbol{y}|\mathfrak{m}^\star)p(\mathfrak{m}^\star)q(\mathfrak{m}|\mathfrak{m}^\star)}{p(\boldsymbol{y}|\mathfrak{m})p(\mathfrak{m})q(\mathfrak{m}^\star|\mathfrak{m})}\right\}, \tag{5}$$

which again involves the marginal likelihoods.

All these problems show the fundamental importance of being able to calculate the marginal likelihood $p(\boldsymbol{y}|\mathfrak{m})$ in Bayesian statistics. At the same time, it can often be very time-consuming to evaluate $p(\boldsymbol{y}|\mathfrak{m})$, especially when the sample size is large. For this reason, a lot of research has been conducted to find ways to reduce this computational burden. When using conjugate priors there exist closed-form expressions for the exact marginal likelihood. Otherwise, it can often be estimated using either simulations or approximation methods [2]. These methods range from MCMC simulation to variations of Laplace approximation [43] and Variational Bayes (VB) [18]. Recently, quite efficient [23] integrated nested Laplace approximations were introduced [37]. Examples of MCMC-based approaches include the harmonic mean estimator [34] and Chib's method [11]. The former estimates the marginal likelihood as the harmonic mean of a sample obtained using MCMC [18]. The latter is based on the Gibbs sampler [19].

Methods using Laplace approximation of the marginal likelihood are particularly relevant when computational time is of importance [18] as they outperform competing methods in run-time by a large factor. Some assumptions have to be made about the posterior, but as we shall see later, the performance remains acceptable in most cases, especially when the sample size is large. Yet, even these approximations can become computationally demanding for increasingly large datasets. The objective of this paper is to propose a novel subsampling strategy for calculating the marginal likelihood. This will result in a more scalable version of Bayesian model selection. More specifically, the proposed methodology will enable using Bayesian model selection and model averaging for generalised linear models (GLM) where a large number of observations is available (so-called *"tall data"*).

We propose a novel general approach based on subsampling optimisation algorithms such as batch stochastic gradient descent (BSGD) or stochastic quasi-Newton (SQN) [9] when calculating the Laplace approximation of the marginal likelihood which is used in MCMC for Bayesian model selection. These algorithms compute the maximum likelihood estimates (MLE) or the posterior modes without using the full dataset at every iteration of the optimisation routine. This ultimately results in a more scalable version of MCMC.

Although the proposed method will work for *any* stochastic optimisation routine, we develop and recommend using a new variant of the BSGD optimisation algorithm, called *subsampling IRLS with stochastic gradient descent* (S-IRLS-SGD), which performs well in practice. The algorithm is available as an R package called `irls.sgd` (https://github.com/jonlachmann/irls.sgd). We also provide some modifications to the `stochQN` package [15] implementing SQN, adaQN [28] and oLBFGS [40] with more efficient implementations of the gradient and Hessian-vector product at https://github.com/jonlachmann/stochQN. These algorithms are also evaluated in this paper but are not recommended due to inferior performance observed in practice for this specific context. Further, we have also created a package called GMJMCMC (https://github.com/jonlachmann/GMJMCMC) which provides an implementation of the mode jumping MCMC (MJMCMC) algorithm [25]. This implementation can be combined with *any* estimator of the marginal likelihood, including SGD, BSGD, S-IRLS-SGD or SQN forming the time-inhomogeneous MCMC (TIMCMC) algorithm that we propose in this paper. All three of these packages are implemented in the R programming language, but computationally intense parts are written in C++ to additionally enhance scalability. These packages provide an implementation of the suggested approach that aims to be ready to use out of the box. Results from extensive simulations as well as real data examples are presented to illustrate the performance of the proposed approach.

The rest of the paper is organised as follows: The class of GLMs and a general standard inferential procedure for Bayesian model selection are presented in Section 2. The suggested approach to combine stochastic optimisation and MCMC through a TIMCMC algorithm is discussed in Section 3. Further, theoretical convergence results and details on the practically recommended choice of stochastic optimisation and MCMC are provided in this section. More specifically, we introduce the S-IRLS-SGD algorithm used for computing the marginal likelihood. In Section 4, we first compare various stochastic optimisation methods on a simulated dataset to show that the recommended S-IRLS-SGD is a reasonable choice in practice. Then, the proposed combination of S-IRLS-SGD and MJMCMC algorithm is applied to several simulated datasets to confirm the theoretical results of the paper. Finally, a real data example is analysed, where full enumeration using IRLS (ground truth) is compared to the approximations obtained after 15000 MJMCMC iterations using IRLS, S-IRLS-SGD and SQN respectively. Section 5 concludes the paper and gives directions for further research. Additional details and examples are provided in Appendices A-F.

2. Model selection for generalised linear models

We will first introduce the notation for our statistical model. Then a description of standard procedures for Bayesian variable selection and their limitations will be presented.

2.1. The model

We consider the following generalised linear model:

$$y_i|\mu_i,\phi \sim \mathfrak{f}(y|\mu_i,\phi), \quad \mu_i = \mathsf{h}^{-1}(\eta_i), \tag{6}$$

$$\eta_i = \beta_0 + \sum_{j=1}^{p} \gamma_j \beta_j x_{ij}. \tag{7}$$

Here, y_i is the response variable while $x_{ij}, j = 1, ..., p$ are the covariates. We assume that $\mathfrak{f}(y|\mu,\phi)$ is a density from the exponential family with corresponding mean μ and dispersion parameter ϕ. Applying the link function $\mathsf{h}(\cdot)$ to μ_i, the specific mean of individual i, yields η_i which is modelled as a linear combination of the covariates. The latent indicators $\gamma_j \in \{0,1\}, j = 1, ..., p$ define if covariate x_{ij} is to be included in the model ($\gamma_j = 1$) or not ($\gamma_j = 0$) while $\beta_j \in \mathbb{R}, j = 0, ..., p$ are the corresponding regression coefficients.

To put the model into a Bayesian framework we have to assign prior distributions to all parameters. Concerning the model structure, we assume

$$\gamma_j|q \sim \text{Binom}(1,q), \quad j = 1, ..., p. \tag{8}$$

Here, q is the prior probability of including a covariate into the model. For the sake of simplicity, prior distributions for different γ_j's are assumed to be independent and identical. The specific choice of priors for $\boldsymbol{\beta}$ and ϕ will be discussed in Section 4 for the specific applications. For posterior distributions we adopt the common notation of conditioning on $\boldsymbol{y} = (y_1, ..., y_n)$, which implicitly also includes conditioning on $\{x_{ij}, i = 1, ..., n, j = 1, ..., p\}$.

Given a set of p covariates, we can define a specific model through the vector $\mathfrak{m} = (\gamma_1, ..., \gamma_p)$. Following the notation used in the Introduction, denote the model space containing 2^p possible models as $\mathcal{M}$. Also for the time being, assume the marginal likelihoods $p(\boldsymbol{y}|\mathfrak{m})$ are available. Our main goal will be Bayesian model selection in which case our interest is in the posterior model probabilities (4).

2.2. MCMC for Bayesian variable selection

If the marginal likelihood (1) can be computed for every model, the exact model posteriors follow from Equation (4). However, for a larger model space, this is often not computationally feasible. Alternatively one can obtain estimates of the posterior probabilities using an MCMC algorithm. For example, the acceptance probabilities from Equation (5) yield the Metropolis-Hastings algorithm, which introduces some proposal distribution $q(\mathfrak{m}^\star|\mathfrak{m})$ on the marginal space of models. Assuming that a set of models $\{\mathfrak{m}^s, s = 1, ..., T\}$ sampled from $p(\mathfrak{m}|\boldsymbol{y})$ is available, the standard Monte Carlo estimator (MC) of the posterior is

$$\hat{p}^{(T)}_{MC}(\mathfrak{m}|\boldsymbol{y}) = T^{-1}\sum_{s=1}^{T} \mathrm{I}(\mathfrak{m}^{(s)} = \mathfrak{m}). \tag{9}$$

Alternatively, the renormalised estimator [RM, 13] of the marginal posterior can be applied:

$$\hat{p}^{(T)}_{RM}(\mathfrak{m}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\mathfrak{m})p(\mathfrak{m})}{\sum_{\mathfrak{m}'\in\mathcal{M}_V^{(T)}} p(\boldsymbol{y}|\mathfrak{m}')p(\mathfrak{m}')} \mathrm{I}(\mathfrak{m} \in \mathcal{M}_V^{(T)}). \tag{10}$$

Here, $\mathcal{M}_V^{(T)}$ denotes the set of all unique models among the sampled models.

The estimates from both of these estimators will, under standard assumptions, converge to the true posterior when the number of iterations $T \to \infty$, but the RM estimates will typically converge faster [13,25].

It is straightforward to see that based on these two estimators we can obtain estimators of the posterior marginal probability (PMP) for any given variable as

$$\hat{p}^{(t)}_{MC}(\gamma_j = 1|\boldsymbol{y}) = \sum_{\mathfrak{m}\in\mathcal{M}_V^{(t)}} \mathrm{I}(\gamma_j = 1)\hat{p}^{(t)}_{MC}(\mathfrak{m}|\boldsymbol{y}) \tag{11}$$

and

$$\hat{p}^{(t)}_{RM}(\gamma_j = 1|\boldsymbol{y}) = \sum_{\mathfrak{m}\in\mathcal{M}_V^{(t)}} \mathrm{I}(\gamma_j = 1)\hat{p}^{(t)}_{RM}(\mathfrak{m}|\boldsymbol{y}). \tag{12}$$

2.3. Mode Jumping MCMC (MJMCMC)

The distribution of models will often be multi-modal. Sampling from a complicated multi-modal distribution using standard MCMC algorithms, such as the Metropolis-Hastings algorithms often leads to problems. A proposal distribution needs to be chosen in a way that allows for both thorough exploration of a nearby mode, as well as the possibility to make transitions to other modes in the model space being explored. Choosing a proposal that makes mostly small steps will explore the current location thoroughly, but almost never escape it. On the other hand, a proposal that suggests large jumps will only rarely find the large jumps to be accepted since the algorithm will end up in models of low probability with respect to the target distribution being explored.

To solve this problem, [44] introduced *mode jumping proposals*, creating the MJMCMC algorithm for continuous variables. Later, [25] adapted it to work for discrete binary variables, used in the case of model selection. Assume that the current state is $\mathfrak{m}$, proposals are then generated in three steps. First, a large jump is made by swapping a large proportion of the binary variables $\gamma_j \to 1-\gamma_j$ with a kernel $q_l(\cdot|\cdot)$, giving $\mathfrak{m}_0^\star$. Second, local optimisation is performed [based on the theory of combinatorial optimisation, 6] with a kernel $q_o(\cdot|\cdot)$, giving $\mathfrak{m}_1^\star$. Finally, a small perturbation of the local optimum is made by swapping a small proportion of the binary variables with a proposal $q_r(\cdot|\cdot)$, giving $\mathfrak{m}^\star$. To be able to calculate the acceptance probability, two backward intermediate states are also calculated to get the reverse path $\mathfrak{m}^\star \to \mathfrak{m}_0 \to \mathfrak{m}_1 \to \mathfrak{m}$. Following the description from [25], this process is described in more detail in Algorithm 1. A proof of convergence was given in [44] for the continuous case and adapted to the discrete case in [25]. The details on the kernels $q_l(\cdot|\cdot)$, $q_o(\cdot|\cdot)$, and $q_r(\cdot|\cdot)$ can also be found in [25].

In [44], the authors note that the mode jumping proposals should only be used a fraction of the time, the rest of the proposals are generated through regular Metropolis-Hastings kernels. They also demonstrate that the algorithm is efficient at exploring complicated target distributions with multiple modes by various examples. In [25], the efficiency in the context of model selection is demonstrated. Yet, the algorithm of [25] does not handle efficiently large data samples and hence can not be applied in such settings. In Section 3, we resolve this issue for a general MCMC algorithm, which will also be valid for MJMCMC as a special case that we use in practice. For the present paper, we reimplemented the MJMCMC algorithm from scratch in a library available at https://github.com/jonlachmann/GMJMCMC with computationally expensive blocks written in C++. To show the validity of the implementation, we reproduced the experiments of [25] with the new implementation. The results of these experiments are presented in Appendix B.

Algorithm 1: One step of the Mode Jumping proposal algorithm moving from current state $\mathfrak{m}$.

```
Sample a large set of components $I \subset \{1, ..., p\}$ uniformly without replacement.
Generate $\mathfrak{m}_0^\star$ by swapping the components $I$ of $\mathfrak{m}$.
Perform a local optimisation defined as $\mathfrak{m}_1^\star \sim q_o(\mathfrak{m}_1^\star|\mathfrak{m}_0^\star)$.
Perform a small randomisation to generate the proposal $\mathfrak{m}^\star \sim q_r(\mathfrak{m}^\star|\mathfrak{m}_1^\star)$.
Reverse the large jump by swapping the components $I$ of $\mathfrak{m}^\star$ to generate $\mathfrak{m}_0$.
Do a backwards optimisation $\mathfrak{m}_1 \sim q_l(\mathfrak{m}_1|\mathfrak{m}_0)$.
Calculate the probability of doing a small randomisation to the starting value $q_r(\mathfrak{m}|\mathfrak{m}_1)$.
Set the next state of the chain to
```

$$\mathfrak{m}' = \begin{cases} \mathfrak{m}^\star & \text{with probability } \alpha = r^\star_{mh}(\mathfrak{m}, \mathfrak{m}^\star; \mathfrak{m}_1, \mathfrak{m}_1^\star); \\ \mathfrak{m} & \text{otherwise,} \end{cases} \tag{13}$$

where

$$r^\star_{mh}(\mathfrak{m}, \mathfrak{m}^\star; \mathfrak{m}_1, \mathfrak{m}_1^\star) = \min\left\{1, \frac{p(\mathfrak{m}^\star|\boldsymbol{y})q_r(\mathfrak{m}|\mathfrak{m}_1)}{p(\mathfrak{m}|\boldsymbol{y})q_r(\mathfrak{m}^\star|\mathfrak{m}_1^\star)}\right\}. \tag{14}$$

2.4. Laplace approximation of the marginal likelihood

The integral from Equation (1) to compute the marginal likelihood $p(\boldsymbol{y}|\mathfrak{m})$ usually does not have a closed-form solution. Computationally expensive calculations of marginal likelihoods become particularly problematic for model selection, where a large number of marginal likelihoods need to be calculated. For this reason, approximations are widely used, most of which are built on asymptotic properties of the posterior [2]. A prime example is the Laplace approximation, which will be central to our subsampling approach.

The Laplace method [29] makes use of the Taylor approximation to obtain approximate solutions of integrals. It is common to apply the Laplace approximation for the likelihood multiplied by the prior for the model parameters $p(\boldsymbol{y}|\boldsymbol{\theta}_\mathfrak{m},\mathfrak{m})p(\boldsymbol{\theta}_\mathfrak{m})$. Let $\tilde{\boldsymbol{\theta}}_\mathfrak{m}$ denote the posterior mode. Then, the approximation of the marginal likelihood looks as follows:

$$\hat{p}(\boldsymbol{y}|\mathfrak{m}) = p(\boldsymbol{y}|\tilde{\boldsymbol{\theta}}_\mathfrak{m},\mathfrak{m})p(\tilde{\boldsymbol{\theta}}_\mathfrak{m}|\mathfrak{m})(2\pi)^{|\boldsymbol{\theta}_\mathfrak{m}|/2}\left|S(\tilde{\boldsymbol{\theta}}_\mathfrak{m},\mathfrak{m})\right|^{-1/2}, \tag{15}$$

where $S(\tilde{\boldsymbol{\theta}}_\mathfrak{m},\mathfrak{m})$ is the Hessian of $\ln[p(\boldsymbol{y}|\tilde{\boldsymbol{\theta}}_\mathfrak{m},\mathfrak{m})p(\tilde{\boldsymbol{\theta}}_\mathfrak{m}|\mathfrak{m})]$ evaluated at the posterior mode $\tilde{\boldsymbol{\theta}}_\mathfrak{m}$ [2].

For most natural priors one can disregard its effect on the posterior for large n. This yields the *Bayesian Information Criterion (BIC)* [41] and a computationally efficient approximation of the marginal posterior. In that case, $\tilde{\boldsymbol{\theta}}_\mathfrak{m}$ is the MLE rather than the posterior mode in (15) and the Hessian $S(\tilde{\boldsymbol{\theta}},\mathbf{m}_\mathfrak{m})$ is also replaced by (minus) the observed Fisher information matrix

$$J_n(\tilde{\boldsymbol{\theta}}_\mathfrak{m},\mathfrak{m}) = -\nabla^2 \ln p(\boldsymbol{y}|\boldsymbol{\theta}_\mathfrak{m},\mathfrak{m})\Big|_{\boldsymbol{\theta}_\mathfrak{m}=\tilde{\boldsymbol{\theta}}_\mathfrak{m}}. \tag{16}$$

Taking the logarithm and removing terms that for large n will be negligible, we get the approximation of the log marginal likelihood $p(\boldsymbol{y}|\mathfrak{m}) \approx p_L(\boldsymbol{y}|\mathfrak{m},\tilde{\boldsymbol{\theta}}_\mathfrak{m})$, where

$$\ln p_L(\boldsymbol{y}|\mathfrak{m},\tilde{\boldsymbol{\theta}}_\mathfrak{m}) = \ln p(\boldsymbol{y}|\tilde{\boldsymbol{\theta}}_\mathfrak{m},\mathfrak{m}) - \frac{\|\boldsymbol{\theta}_\mathfrak{m}\|_0}{2}\ln n. \tag{17}$$

Here, $\|\boldsymbol{\theta}_\mathfrak{m}\|_0$ is the l_0 norm for the dimension of $\boldsymbol{\theta}_\mathfrak{m}$, i.e. the norm is counting the number of non-zero parameters. Asymptotically, the logarithms of (15) and (17) are equivalent. In the following, we will use the latter unless analytical marginal likelihoods are available. The approximation (15) becomes exact for a model with Gaussian observations and a normal prior for the coefficients $\boldsymbol{\beta}$ since the log of the likelihood for the observations is then taking a negative quadratic form; otherwise, it will have an error of order $\mathcal{O}(n^{-1})$. The numerical optimisation of $\tilde{\boldsymbol{\theta}}_\mathfrak{m}$ is usually performed via second-order optimisation methods like *iterative reweighted least squares (IRLS)* [5,39], where a *weighted least squares (WLS)* equation is iteratively reweighted with the inverse of residuals from the previous solutions. These methods involve using all data at each iteration. The computational complexity is dominated by the matrix multiplication and matrix inversion in the WLS step which has complexity $\mathcal{O}(\max(m^2n, m^3))$, where n is the number of observations and m is the number of parameters to be estimated. Hence, for a dataset with many observations, this complexity will be dominated by $\mathcal{O}(m^2n)$ which grows linearly with the number of observations and quadratically in the number of parameters. For large datasets, this procedure often becomes too computationally demanding.

Using the Laplace approximation, we will turn our focus on the approximate model probabilities given by

$$p_L(\mathfrak{m}|\boldsymbol{y}) = \frac{p(\mathfrak{m})p_L(\boldsymbol{y}|\mathfrak{m},\tilde{\boldsymbol{\theta}}_\mathfrak{m})}{\sum_{\mathfrak{m}'\in\mathcal{M}} p(\mathfrak{m}')p_L(\boldsymbol{y}|\mathfrak{m}',\tilde{\boldsymbol{\theta}}'_\mathfrak{m})}. \tag{18}$$

In the next section, we will construct MCMC algorithms for sampling from $p_L(\mathfrak{m}|\boldsymbol{y})$ in case of tall data, where the computation of $\tilde{\boldsymbol{\theta}}_\mathfrak{m}$ is getting difficult.

3. A time-inhomogeneous MCMC algorithm

The main novelty of our procedure will be to handle the problem that even the posterior mode $\tilde{\boldsymbol{\theta}}_\mathfrak{m}$ will be difficult to compute when the sample size is large. To resolve this, a novel combination of Markov chain Monte Carlo and iterative procedures for updating estimates of the posterior modes is proposed. In the first part of Section 3, we introduce a very general algorithm for combining MCMC (including MJMCMC) and stochastic optimisation. Then, in Section 3.1, we provide general theoretical results for this type of algorithm. Finally, in Section 3.2, the general algorithm is linked to specific stochastic optimisation methods.

The suggested approach is directly linked to the general idea of simulating from a time-inhomogeneous MCMC [16] allowing to rely upon the theory developed for TIMCMC [16,17,38]. Our TIMCMC for each time step has the following target:

$$p_L^{(t)}(\mathfrak{m}|\boldsymbol{y}) = \frac{p(\mathfrak{m})p_L(\boldsymbol{y}|\mathfrak{m},\tilde{\boldsymbol{\theta}}^{(t)}_\mathfrak{m})}{\sum_{\mathfrak{m}'} p(\mathfrak{m}')p_L(\boldsymbol{y}|\mathfrak{m}',\tilde{\boldsymbol{\theta}}^{(t)}_{\mathfrak{m}'})}. \tag{19}$$

Here, $\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{(t)}$ is some estimate of $\tilde{\boldsymbol{\theta}}_{\mathfrak{m}}$ at time t. In the following, we will assume we have a procedure generating a sequence $\{\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{t}, t = 0, 1, 2, ...\}$ such that

$$\lim_{t\to\infty} \hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{t} \to \tilde{\boldsymbol{\theta}}_{\mathfrak{m}} \quad \text{either (a) almost surely or (b) in probability for any } \mathfrak{m} \in \mathcal{M}. \tag{20}$$

Algorithm 2 describes the full procedure. Here, `init-mode()` is a procedure which for a given model $\mathfrak{m}$ finds an initial estimate $\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{0}$ while `update-mode()` is a procedure which updates the estimate given the current value.

Algorithm 2: The TIMCMC algorithm.

```
Select a model $\mathfrak{m}^{(0)}$, put $\hat{\boldsymbol{\theta}}_{\mathfrak{m}} = \texttt{init-mode}(\mathfrak{m}^{(0)})$ and $\mathcal{M}_V = \{\mathfrak{m}^{(0)}\}$;
for $t = 1, 2, ...$ do
    Select a model $\mathfrak{m}^\star$ from some proposal distribution $q(\mathfrak{m}^\star|\mathfrak{m}^{(t-1)})$;
    if $\mathfrak{m}^\star \notin \mathcal{M}_V$ then
        Put $\hat{\boldsymbol{\theta}}_{\mathfrak{m}^\star} \leftarrow \texttt{init-mode}(\mathfrak{m}^\star)$ and $\mathcal{M}_V \leftarrow \{\mathcal{M}_V, \mathfrak{m}^\star\}$;
    else
        Put $\hat{\boldsymbol{\theta}}_{\mathfrak{m}^\star} \leftarrow \texttt{update-mode}(\mathfrak{m}^\star, \hat{\boldsymbol{\theta}}_{\mathfrak{m}^\star})$;
    end
    Set the next state of the chain to
        $\mathfrak{m}^t = \begin{cases} \mathfrak{m}^\star & \text{with probability } r^\star_{mh}(\mathfrak{m}^{t-1}, \mathfrak{m}^\star); \\ \mathfrak{m}^{t-1} & \text{otherwise;} \end{cases}$   (21)
     where
        $r^\star_{mh}(\mathfrak{m}, \mathfrak{m}^\star) = \min\left\{1, \frac{p(\mathfrak{m}^\star)p_L(\boldsymbol{y}|\mathfrak{m}^\star, \hat{\boldsymbol{\theta}}_{\mathfrak{m}^\star})q(\mathfrak{m}|\mathfrak{m}^\star)}{p(\mathfrak{m})p_L(\boldsymbol{y}|\mathfrak{m}, \hat{\boldsymbol{\theta}}_{\mathfrak{m}})q(\mathfrak{m}^\star|\mathfrak{m})}\right\}$,   (22)
     where $q(\cdot|\cdot)$ is the proposal of the addressed MCMC.
end
```

Remark 1. If `init-mode()` obtains the posterior mode directly and no further updates are made by `update-mode()`, the algorithm reduces to an ordinary Metropolis-Hastings algorithm with $p_L(\mathfrak{m}|\boldsymbol{y})$ as target distribution.

Remark 2. Sometimes the use of auxiliary variables for the generation of $\mathfrak{m}^\star$ can be beneficial. Examples are the pseudo-marginal MCMC [3] and MJMCMC (Section 2.3). In such cases, the main structure of the algorithm can still be used, but the acceptance probabilities need to be appropriately modified, e.g. $q(\cdot|\cdot)$ will be $q_r(\cdot|\cdot)$ from Algorithm 1.

3.1. Theoretical properties

In this section, we will derive general conditions under which Algorithm 2 will have the appropriate convergence properties. Although the algorithm is of interest for many types of update procedures, we will consider here only those where the updating always increases the (log-)likelihood value [42]. We will later see how this can be easily achieved in practice.

Theorem 1. *Assume $\mathcal{M}$ is finite with $p(\mathfrak{m}) > 0$ $\forall \mathfrak{m} \in \mathcal{M}$ and $\sum_{\mathfrak{m}\in\mathcal{M}} p(\mathfrak{m}) = 1$. Assume further that for any $\mathfrak{m} \in \mathcal{M}$ `update-mode()` generates a sequence $\{\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{t}\}$ such that $p_L(\boldsymbol{y}|\mathfrak{m}, \hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{t}) \geq p_L(\boldsymbol{y}|\mathfrak{m}, \hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{t-1}) > 0$ and the sequence converges (a) almost surely $\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{t} \xrightarrow{a.s.} \hat{\boldsymbol{\theta}}_{\mathfrak{m}}$ or (b) in probability $\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{t} \xrightarrow{p} \tilde{\boldsymbol{\theta}}_{\mathfrak{m}}$. Assume the proposal distribution $q(\cdot|\cdot)$ generates an irreducible Markov chain on $\mathcal{M}$. Then*

(I) *the Markov chain $\{\mathfrak{m}^t\}$ induced by Algorithm 2 is irreducible and recurrent;*
(II) *for the renormalised estimate (10) it holds that*
  (a) $\hat{p}_{RM}^{(t)}(\mathfrak{m}|\boldsymbol{y}) \xrightarrow{a.s.} p_L(\mathfrak{m}|\boldsymbol{y})$ *as $t \to \infty$*
  (b) $\hat{p}_{RM}^{(t)}(\mathfrak{m}|\boldsymbol{y}) \xrightarrow{p} p_L(\mathfrak{m}|\boldsymbol{y})$.

Proof. Part (I) Define $p_0 = \min_{\mathfrak{m}\in\mathcal{M}} p_L(\boldsymbol{y}|\mathfrak{m}, \hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{(0)})$ and $p_\infty = \max_{\mathfrak{m}\in\mathcal{M}} p_L(\boldsymbol{y}|\mathfrak{m}, \tilde{\theta}_{\mathfrak{m}})$. For the specific classes of data distributions considered, we have that $p_0 > 0$ and $p_\infty < \infty$. Then

$$0 < p_0 \leq p_L(\boldsymbol{y}|\mathfrak{m}, \hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{(0)}) \leq p_L(\boldsymbol{y}|\mathfrak{m}, \hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{(t)}) \leq p_L(\boldsymbol{y}|\mathfrak{m}, \tilde{\boldsymbol{\theta}}_{\mathfrak{m}}) \leq p_\infty < \infty$$

which gives

$$p^t(\mathfrak{m}|\boldsymbol{y}) \equiv \frac{p(\mathfrak{m})p_(\boldsymbol{y}|\mathfrak{m},\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{(t)})}{\sum_{\mathfrak{m}}' p(\mathfrak{m}')p_L(\boldsymbol{y}|\mathfrak{m}',\hat{\boldsymbol{\theta}}_{\mathfrak{m}'}^{(t)})} \geq \frac{p(\mathfrak{m})p_0}{\sum_{\mathfrak{m}}' p(\mathfrak{m}')p_\infty} = \frac{p_0}{p_\infty}p(\mathfrak{m}),$$

showing that there is a lower bound for the model probabilities related to the acceptance probabilities given by (22). Combined with the assumptions about the proposal distribution $q(\cdot|\cdot)$, this implies irreducibility. Since the space of states is discrete and finite, the Markov chain will also be recurrent.

Part (II) By recurrency, every model will be visited an infinite number of times as $t \to \infty$. By the continuous mapping theorem [30], as $t \to \infty$ we have that $p_L(\boldsymbol{y}|\mathfrak{m},\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{(t)}) \xrightarrow{a.s.} p_L(\boldsymbol{y}|\mathfrak{m},\tilde{\boldsymbol{\theta}}_{\mathfrak{m}})$. Combining this with the fact that $\mathcal{M}$ is finite and following Bayes theorem and properties of convergence of the sums and products of converging sequences as $t \to \infty$ we get for (a) that $\hat{p}_{RM}^{(t)}(\mathfrak{m}|\boldsymbol{y}) \xrightarrow{a.s.} p_L(\mathfrak{m}|\boldsymbol{y})$ and for (b) that $\hat{p}_{RM}^{(t)}(\mathfrak{m}|\boldsymbol{y}) \xrightarrow{p} p(\mathfrak{m}|\boldsymbol{y})$, where $\hat{p}_{RM}^{(t)}(\mathfrak{m}|\boldsymbol{y})$ are obtained through Equation (10) with $\hat{p}_L(\boldsymbol{y}|\mathfrak{m},\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{(t)})$ plugged in for $p(\boldsymbol{y}|\mathfrak{m})$. □

A general result for MC estimates is available by applying the general theory about time-inhomogeneous Markov chains considered in [17]. An important criterion is their assumption A3 which is translated to our setting below:

Condition 1. *Assume*

1. *for all $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\mathfrak{m}}, \mathfrak{m} \in \mathcal{M}\} \in \boldsymbol{\Theta}$ the transition matrix $P_{\boldsymbol{\theta}}$ induced by the Markov chain in Algorithm 2 is irreducible and aperiodic for all $\boldsymbol{\theta} \in \Theta$;*
2. *there exists a function $V : \mathcal{M} \to [1, +\infty)$ such that for any $\boldsymbol{\theta} \in \Theta$ there exist some constants $b_{\boldsymbol{\theta}} < \infty$, $\delta_{\boldsymbol{\theta}} \in (0,1)$, $\lambda_{\boldsymbol{\theta}} \in (0,1)$ and a probability distribution $\nu_{\boldsymbol{\theta}}$ on $\mathcal{M}$ such that*

$$\sum_{\mathfrak{m}'\in\mathcal{M}} P_{\boldsymbol{\theta}}(\mathfrak{m}'|\mathfrak{m})V(\mathfrak{m}') \leq \lambda_{\boldsymbol{\theta}} V(\mathfrak{m}) + b_{\boldsymbol{\theta}} \qquad \textit{for all } \mathfrak{m} \in \mathcal{M}; \tag{23}$$

$$P_{\boldsymbol{\theta}}(\mathfrak{m}'|\mathfrak{m}) \geq \delta_{\boldsymbol{\theta}} \nu_{\boldsymbol{\theta}}(\mathfrak{m}') \mathrm{I}\left(V(\mathfrak{m}) \leq \tfrac{2b_{\boldsymbol{\theta}}}{1-\lambda_{\boldsymbol{\theta}}} - 1\right) \qquad \textit{for all } \mathfrak{m}, \mathfrak{m}' \in \mathcal{M}. \tag{24}$$

Theorem 2. *Assume the conditions of Theorem 1 and Condition 1 hold. Then Monte Carlo estimates of model probabilities based on the frequencies of a MCMC (MJMCMC) algorithm will almost surely converge to the true posteriors $\hat{p}_{MC}^{(t)}(\mathfrak{m}|\boldsymbol{y}) \xrightarrow{a.s.} p_L(\mathfrak{m}|\boldsymbol{y})$ as $t \to \infty$.*

Proof. $\mathcal{M}$ is a finite discrete countable space of models, which is completely metrizable and separable. Hence, it is a Polish space. Combined with Condition 1, the results follow from Theorem 2.11 in [17]. □

Remark 3. The first part of Condition 1 is directly fulfilled under the regularity conditions of Theorem 1. For the second part, by defining $V(\mathfrak{m}) \equiv 1$, inequality (23) will also be directly fulfilled for $b_{\boldsymbol{\theta}} \geq 1$ and for all $\lambda_{\boldsymbol{\theta}} \in (0,1)$. Further, if $\boldsymbol{\Theta}$ is limited to $\boldsymbol{\Theta} = \{\boldsymbol{\theta} : p_L(\boldsymbol{y}|\mathfrak{m},\boldsymbol{\theta}_{\mathfrak{m}}) \geq p_L(\boldsymbol{y}|\mathfrak{m},\hat{\boldsymbol{\theta}}_{\mathfrak{m}}^{(0)})\}$ (which is satisfied by design of `update-mode()`) and if $q(\mathfrak{m}'|\mathfrak{m}) > 0$ for all $\mathfrak{m}, \mathfrak{m}' \in \mathcal{M}$ (which is a reasonable MCMC proposal), then there will be a lower positive limit $\underline{P} > 0$ of $P_{\boldsymbol{\theta}}(\mathfrak{m}'|\mathfrak{m})$ and also equation (24) will be fulfilled by choosing $\nu_{\boldsymbol{\theta}}(\mathfrak{m}) = 1/|\mathcal{M}|$ and $\delta_{\boldsymbol{\theta}}$ small enough and $\forall \lambda_{\boldsymbol{\theta}} \in (0,1)$. More general choices of proposals $q(\mathfrak{m}'|\mathfrak{m})$ should also be valid with proper choices of $V(\cdot)$ and the constants involved, but will not be pursued further here.

3.2. *Constructing the* `update-mode()`

We have a theoretical requirement for the `update-mode()` imposed by Theorem 1 in that we have a method generating a sequence of parameters converging either (a) almost surely or (b) in probability. A standard IRLS will converge surely and thus automatically satisfies both (a) and (b). But we are interested in using subsampling based stochastic optimisation, for which convergence might be in general more difficult to obtain.

Assume `opt`$(\hat{\boldsymbol{\theta}}^{\star})$ to be *any* iterative optimisation routine starting at $\hat{\boldsymbol{\theta}}^{\star}$. In Algorithm 3, we provide a very general approach based on generating a random variable $\boldsymbol{\varepsilon}^{(t)}$ after optimisation `opt`$(\check{\boldsymbol{\theta}}^{(t)})$ for each step t.

Algorithm 3: Randomisation after stochastic optimisation.

```
Initialise $\check{\boldsymbol{\theta}}^{(0)}$ and put $\hat{\boldsymbol{\theta}}^{(0)} = \mathtt{opt}(\check{\boldsymbol{\theta}}^{(0)})$
for $t = 1, 2, \ldots$ do
    Initialise $\check{\boldsymbol{\theta}}^{(t)}$ and generate a random $\boldsymbol{\varepsilon}^{(t)}$, put $\check{\boldsymbol{\theta}}^{(t)} = \mathtt{opt}(\check{\boldsymbol{\theta}}^{(t)}) + \boldsymbol{\varepsilon}^{(t)}$ and
        $\hat{\boldsymbol{\theta}}^{(t)} = \begin{cases} \check{\boldsymbol{\theta}}^{(t)} & \text{if } p_L(\boldsymbol{y}|\mathfrak{m},(\check{\boldsymbol{\theta}}^{(t)}) \geq p_L(\boldsymbol{y}|\mathfrak{m},(\hat{\boldsymbol{\theta}}^{(t-1)}); \\ \hat{\boldsymbol{\theta}}^{(t-1)} & \text{otherwise.} \end{cases}$
end
```

This algorithm falls within the general class of random search techniques [31,42] for which convergence in probability is guaranteed under very mild conditions, for instance, if the random generator for $\breve{\boldsymbol{\theta}}^{(t)}$ gives zero probability for repeatedly missing any set A of positive volume. This can easily be achieved by allowing $\boldsymbol{\varepsilon}^{(t)}$ to have a positive density on the whole parameter space. For example, $\boldsymbol{\varepsilon}^{(t)}$ can be chosen to be $\Gamma \times \mathrm{N}(0, \sigma^2_{\text{rand}})$, where $\Pr(\Gamma = 1) = p_{\text{rand}} = 1 - \Pr(\Gamma = 0)$ and both σ^2_{rand} and p_{rand} can be selected to be very small in practice to not deviate drastically from the results produced by stochastic optimisation. With such $\boldsymbol{\varepsilon}^{(t)}$, Algorithm 3 easily satisfies the regularity condition of convergence in probability of the parameters required in Theorem 1 allowing to guarantee convergence of the renormalised estimates of posterior probabilities to the target. Now, we shall briefly discuss the possibilities for choosing `opt()`, whilst the detailed description of them is given in Appendix A.

The most well-known optimisation algorithm exploiting a subsampling approach that can be used as `opt()` is *stochastic gradient descent (SGD)*. According to [36], the SGD optimisation algorithm asymptotically converges to a maximum of a concave function. SGD is a first-order optimisation method, which uses an unbiased estimate of the gradient of the objective function based on just one observation of the data to decide in which direction a step will be taken. If one uses several random observations to compute an unbiased estimate of the gradient of the objective function, the algorithm is referred to as batch SGD (BSGD) which is another choice of `opt()`. If one uses a full sample, the algorithm collapses to a regular gradient descent (GD) for which the true gradient is used. GD is not stochastic and as such is not of interest to be used as `opt()`. For these first-order optimisation algorithms with an appropriately chosen step size, the rate of convergence is generally Q-linear [32] for a function with a sufficiently smooth gradient.

One of the most important advances in the field of nonlinear optimisation has been the emergence of quasi-Newton methods [8]. These methods build on approximations of the Hessian using only information about the gradient and are applicable to both convex and non-convex problems. Versions that scale well with the number of variables such as limited memory methods have proven to be effective in a wide range of applications where the number of variables can be in the millions. However, in this paper, we are rather concerned with settings where n is large. For this situation, extensions of quasi-Newton methods to the stochastic situation, allowing for subsampling from the data, were proposed in [15,28,40]. This resulted in the emergence of stochastic quasi-Newton (SQN) optimisation. SQN methods rely on approximations of both the gradient and the Hessian of the objective function using subsampling from the data. Thus, these methods may be seen as advances in second-order stochastic optimisation, although according to [8], they still have only a sub-linear rate of convergence [32]. This does not give obvious theoretical advantages as compared to SGD or BSGD optimisation. In Appendix A, a classical SQN algorithm [15] as well as an adaptive quasi-Newton (adaQN) [28], and an online limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm (oLBFGS) [40] are described. All three of them, in principle, can be used as `opt()` in Algorithm 3. We also updated the existing R library `stochQN` for these methods with calculations of the estimates of the gradient and (if required) Hessian-vector product in C++. This allowed improving the speed of SQN optimisation in R. Our improved package implementation is publicly available on GitHub at https://github.com/jonlachmann/stochQN. Also, if SGD/BSGD are used directly as `update-mode()` with no randomisation used, the theoretical almost sure convergence of both RM and MC estimates is valid. Technical details are given in Appendix A.

Even though we have a theoretical guarantee that Algorithm 3 will work in the asymptotic limit for *any* stochastic optimisation method, none of the standard stochastic optimisation methods allowed us in practice to achieve acceptable performance of TIMCMC within a *limited* number of iterations. In the following subsection, we describe an initialisation scheme for SGD and BSGD based on subsampling IRLS (S-IRLS) which resulted in good practical performance. However, it is important to note that there exist more recent advances in stochastic second-order methods available in the literature that might further improve the performance of TIMCMC when used as `opt()` in `update-mode()`. More specifically, the LISSA method suggested in [1] holds a strong promise for being used as our `opt()`. However, we leave testing it for further research due to the absence (to the best of our knowledge) of relevant R, C++ or C implementations that could be efficiently integrated into our TIMCMC algorithm.

3.2.1. *Subsampling IRLS with BSGD*

The main drawback of SGD and BSGD is that the number of iterations required for convergence may depend crucially on the quality of their starting points. If starting in a bad region of the parameter space, which is not unlikely for randomised starting points, it might take SGD, and BSGD prohibitively long to converge, making them of little use for our TIMCMC in practice. Also, the main problem of SQN methods is that the approximate Hessian becomes quite unstable for highly correlated data, making them perform poorly in such cases. To resolve the issues above, we propose an algorithm that obtains the starting points for SGD using a Subsampling IRLS (S-IRLS) algorithm. S-IRLS is a standard IRLS procedure [5,39], where the weights in a *weighted least squares* equation are iteratively reweighted with the inverse of residuals from the previous solutions and most of the weights at each iteration are randomly set to zero. This corresponds to that only a subsample of data is used at each iteration. After S-IRLS is run for a number of iterations, SGD or BSGD is run for more stable convergence. The overall idea is that S-IRLS, being a quasi-Newton method, is able to quickly provide a rough estimate of the parameters. However, due to the in some cases unstable Hessian estimates, SGD (BSGD) is then run with the S-IRLS estimate as a starting point to practically and theoretically guarantee the convergence to the mode. The resulting algorithm is abbreviated as S-IRLS-SGD in what follows.

In S-IRLS, at each iteration a random subsample is drawn with selection probabilities proportional to $\boldsymbol{w} + \boldsymbol{\varepsilon}_w$, where $\boldsymbol{w}$ is the current weight vector and $\boldsymbol{\varepsilon}_w$ is used to regularise the sampling procedure and make sure that each observation can

Algorithm 4: Subsampling Iteratively Reweighted Least Squares (S-IRLS). Subscript $_S$ to a vector relates to the sub-vector with elements corresponding to the subsample S.

```
Initialise the regression coefficients $\hat{\boldsymbol\beta}_0$ and put $w_i = 1$ for all $i$.
Select a random subsample $\mathcal{S} \subset \{1, \dots, n\}$ of size $b$.
Calculate the linear predictor $\hat{\boldsymbol\eta}_S = \boldsymbol{x}_{\mathcal{S}}^T \hat{\boldsymbol\beta}_0$ and mean vector $\hat{\boldsymbol\mu}_S = \mathsf{h}^{-1}(\hat{\boldsymbol\eta}_S)$.
for $v = 1, 2, \dots, V$ do
    Calculate $\boldsymbol\xi_S = \frac{d\mathsf{h}^{-1}(\boldsymbol\eta_S)}{d\boldsymbol\eta_S}\Big|_{\boldsymbol\eta_S = \hat{\boldsymbol\eta}_S}$, $\boldsymbol{z}_S = \hat{\boldsymbol\eta}_S + (\boldsymbol{y}_{\mathcal{S}} - \hat{\boldsymbol\mu}_S)/\boldsymbol\xi_S$ and $\sigma^2_{\boldsymbol\mu,S} = \mathrm{Var}(\boldsymbol{y}_{\mathcal{S}})$.
    Update the weights for the current subsample $\boldsymbol{w}_{\mathcal{S}} = \left(\boldsymbol\xi_S^2/\sigma^2_{\boldsymbol\mu,S}\right)^{0.5}$ and put $\boldsymbol{W}_S = \mathrm{Diag}(\boldsymbol{w}_{\mathcal{S}})$.
    Update the values of $\hat{\boldsymbol\beta}_v$ according to the cooling schedule

        $\hat{\boldsymbol\beta}_v = \tau_v \hat{\boldsymbol\beta}_v^\star + (1 - \tau_v)\hat{\boldsymbol\beta}_{v-1}$        (25)

    and $\hat{\boldsymbol\beta}_v^\star = (\boldsymbol{x}_{\mathcal{S}}^T \boldsymbol{W}_{\mathcal{S}} \boldsymbol{x}_S)^{-1} \boldsymbol{x}_{\mathcal{S}}^T \boldsymbol{W}_{\mathcal{S}} \boldsymbol{z}_S$.
    Select a new subsample $\mathcal{S} \subset \{1, \dots, n\}$ of size $b$ with probabilities proportional to $w_i + \boldsymbol\varepsilon_w$, where $\boldsymbol{w}$ is the current vector of weights.
    Calculate the new linear predictor $\hat{\boldsymbol\eta}_S = \boldsymbol{x}_{\mathcal{S}} \hat\beta_v$ and mean vector $\hat{\boldsymbol\mu}_S = \mathsf{h}^{-1}(\hat{\boldsymbol\eta}_S)$.
    Calculate the estimated deviance $d_v$ based on the current subsample $\mathcal{S}$.
    if $(d_v - d_{v-1})/|d_{v-1}| > \delta_{expl}$ then
        Reset the values of $\hat{\boldsymbol\beta}_v$ to those from two iterations ago and halve the temperature:

            $\hat{\boldsymbol\beta}_v = \hat{\boldsymbol\beta}_{v-2}, \quad \tau_v = 0.5\tau_v.$        (26)

        Recalculate the new linear predictor $\hat\eta_S = \boldsymbol{x}_{\mathcal{S}}^T \hat{\boldsymbol\beta}_v$ and mean vector $\hat{\boldsymbol\mu}_S = \mathsf{h}^{-1}(\hat{\boldsymbol\eta}_S)$.
    end
end
```

be sampled. This way, we can estimate the expected information matrix $\boldsymbol{x}^T \mathrm{diag}(\boldsymbol{w})\boldsymbol{x}$ on a subsample and the complexity is reduced to $\mathcal{O}(\max(m^2 b, m^3))$, where b is the size of our subsample. The algorithm is described formally as Algorithm 4. In this algorithm, one needs to set a number of tuning parameters for optimal performance, these are

- b: size of subsamples
- V: total number of iterations
- $\tau_0, r, v_{\text{const}}$: three parameters which determine the temperature τ_v in equation (27)
- δ_{expl}: threshold on the relative increase of deviance

The temperature τ_v which is used to improve the convergence properties of the algorithm determines the rate at which $\boldsymbol\beta$ is allowed to change between iterations, see Equation (27). We have chosen to use what is called a constant - exponential decay cooling schedule, which means that the temperature is kept constant for some initial iterations determined by v_{const}, after which it is subject to exponential decay

$$\tau_v = \tau_0 \cdot r^{\max(v - v_{\text{const}}, 0)}. \tag{27}$$

This kind of cooling is also found in some SGD implementations to homogenise the iterative solutions and avoid an unreasonably large impact from an unfortunate subsample. The variances $\sigma^2_{\boldsymbol\mu,S}$ needed for the calculation of the weights are obtained through standard formulas related to generalised linear models. Note that these variances are only needed up to a proportionality constant, making the dispersion parameter ϕ in (6) unnecessary to specify.

S-IRLS is able to quickly approach the vicinity of the optimal solution even when the starting point is far from it. However, when approaching it, the Hessian estimates may become rather unstable. This problem is taken into account by detecting what we call *exploding deviance*, which we define to be the case when the estimated deviance increases by a factor larger than δ_{expl}. This happens more often the smaller the size of the subsample because then the estimate of the Hessian will be more unstable. If this occurs, following popular IRLS implementations for GLM from the R library `fastglm`, the current estimate $\boldsymbol\beta$ is set to $\boldsymbol\beta_{t-2}$ and the temperature is halved to avoid a loop of deviance explosions and backtracking. The number of S-IRLS iterations, V, is one of the tuning parameters. It can be set rather small, i.e. below 100. If $V = 0$, a random initialisation is used, and S-IRLS-SGD collapses to either SGD or BSGD depending on the subsample size chosen. The implementation of SGD, BSGD, and S-IRLS-SGD with efficient calculations of computationally intensive parts written in C++ is available on GitHub at https://github.com/jonlachmann/irls.sgd.

4. Experiments

In this section, we first provide in Example 1 an empirical evaluation of the performance of IRLS, BSGD, SQN, adaQN, oLBFGS, and S-IRLS-SGD to motivate the use of S-IRLS-SGD in the following simulation-based examples. Then, in Example 2, we evaluate how S-IRLS-SGD based on different sub-sample sizes with multiple revisits of each model works for estimating the marginal posterior inclusion probabilities under full enumeration with a controlled number of revisits of each model. In

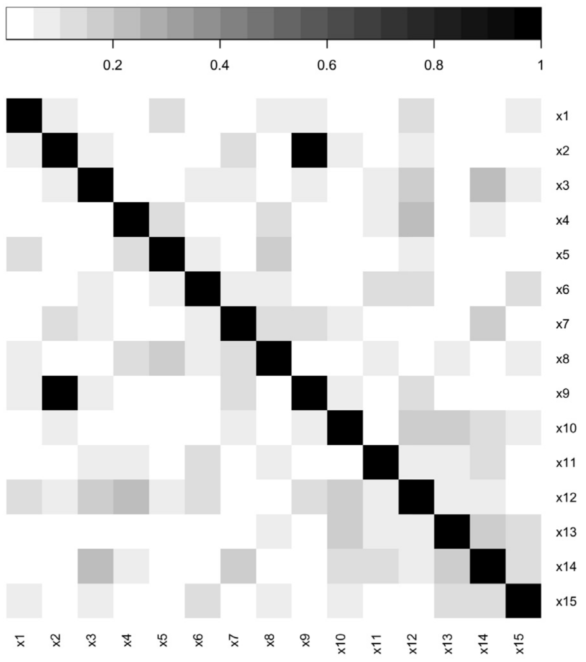

Fig. 1. Absolute correlation between the exogenous x variables for the data used in Examples 1 and 2. The two variables x_2, x_9 are highly correlated, with the remaining variables having low to moderate correlation.

Example 3, we show how S-IRLS-SGD works in combination with MJMCMC on several simulated datasets of different sizes and with different probability distributions for the responses. Finally, in Example 4, we analyse a real dataset and compare the performance of IRLS, S-IRLS-SGD, and SQN.

For Examples 1-3, several simulated datasets were generated. The datasets consist of one dependent variable $\boldsymbol{y}$, and 15 independent variables $\boldsymbol{x}$. We explored the cases with n equal to 10 000 and 100 000 observations. By combining the independent variables in all possible ways we can create $2^{15} = 32\,768$ different linear models.

The data were generated in a way to be comparable to the examples in [14] and [23]. This means that we used the same covariance matrix for the independent variables $\boldsymbol{x}$, and a vector of parameters proportional to $\boldsymbol{\beta} = (0.48, 8.72, 1.76, 1.87, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0)$ when generating the dependent variable $\boldsymbol{y}$. The covariance structure of the data is illustrated in Fig. 1. The coefficients $\boldsymbol{\beta}$ of the data generating model were adjusted to sample size n as follows:

$$\boldsymbol{\beta}^{\star} = \frac{\boldsymbol{\beta}}{\sqrt{n/100}}\,. \tag{28}$$

This will give approximately the same signal strength in the data as $\boldsymbol{\beta}$ did for 100 observations in [26]. The dependent variables $\boldsymbol{y}$ for n observations was generated according to the following distribution:

$$y_i \sim \mathrm{N}(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}^{\star}, 1). \tag{29}$$

Data for logistic regression models were generated using the same covariates as for Gaussian models, creating dependent variables $y_i{}^{\star}$ according to

$$y_i{}^{\star} \sim \text{Bernoulli}\left(\frac{1}{1+\exp\left(-(y_i - \bar{y})\right)}\right). \tag{30}$$

Here, $\bar{y}$ is the sample mean of y_i's with $i \in \{1, \dots, n.\}$. Thus, we generated 4 datasets in total that are used in Examples 2 and 3.

For all experiments, Zellner's g-prior [47] was used for the coefficients with the parameter

$$g = \frac{100}{\sqrt{n/100}}.$$

This choice empirically ensures that a single model will not contain most of the posterior mass, avoiding the PMPs to collapse to either 0 or 1. This is important to ensure that the true PMPs are challenging to estimate, making it possible to examine how well different algorithms are performing. Also, in all the experiments, the prior inclusion probability q was set to 0.5 for all covariates.

Table 1
Hyper-parameters of the compared algorithms, computational costs per iteration and their expected computational costs. Here, a model with $p = 15$ resulting in 16 parameters is assumed and the sample size n is 10^6 observations. Decay and α_0 are the tuning parameters of exponential cooling for the sequence of step sizes α_t for the BSGD optimisation routines [7]. $\star$ The step size at time t is given by $\alpha_0 \cdot 1/(\sqrt{t/100} + 1)$.

	Iterations	% of n	$p + 1$	decay	α_0	Cost per step	Full cost
IRLS	5×10^0	100%	16	-	-	2.56×10^8	1.28×10^9
BSGD 500	5×10^2	0.10%	16	0.99995	0.20	1.6×10^4	8×10^6
BSGD 1K	1×10^3	0.10%	16	0.99995	0.20	1.6×10^4	1.6×10^7
BSGD 5K	5×10^3	0.10%	16	0.99995	0.20	1.6×10^4	8×10^7
BSGD 10K	1×10^4	0.10%	16	0.99995	0.20	1.6×10^4	1.6×10^8
S-IRLS	7.5×10^1	0.10%	16	-	-	2.56×10^5	1.92×10^7
SGD	5×10^2	0.10%	16	0.99000	0.05	1.6×10^4	8×10^6
S-IRLS-SGD	5.75×10^2	0.10%	16	-	-	-	2.72×10^7
SQN	2.5×10^3	0.10%	16	$\star$	0.10	1.6×10^4	4×10^7
oLBFGS	2.5×10^3	0.10%	16	$\star$	0.10	1.6×10^4	4×10^7
adaQN	2.5×10^3	0.10%	16	$\star$	0.10	1×10^3	2.5×10^6

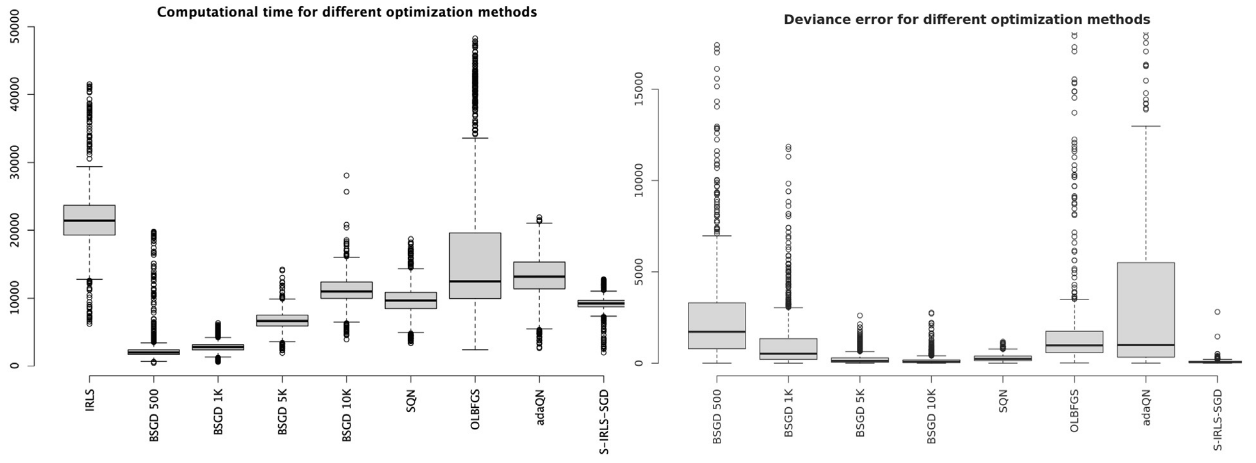

Fig. 2. Boxplots of total computational time (left) and errors of estimated deviances (right) for different optimisation algorithms and tuning parameters from Table 1. For four best ones see also Fig. C.13 in the Appendix C of the article, which gives a higher resolution.

4.1. Example 1: simulated data (stochastic optimisation)

In the first experiment, the simulated dataset with the binary dependent variable, and 1 million observations was used. For this example, a full enumeration of all the 32 768 possible models was carried out using IRLS and the 128 models with the highest posterior mass were selected. These models were then re-evaluated using different optimisation algorithms of interest. 20 runs per model for each setting of each of the algorithms were performed. These algorithms and their tuning parameters are presented in Table 1. This table also lists theoretical computational costs for the addressed methods and tuning parameters.

In Fig. 2, we present boxplots of computational time (left) and errors of estimated deviance (right). As expected from the theoretical complexities of the algorithms in Table 1, the addressed setting of S-IRLS-SGD initialisation gives a reasonable computational time compared to running BSGD, SQN, adaQN, and oLBFGS or using the full IRLS. Furthermore, the deviations of estimated deviance from the true one are the smallest for S-IRLS-SGD. For this reason, we shall focus on the S-IRLS-SGD variant of BSGD in the simulation experiments to follow. The second best stochastic method that might be of interest is SQN according to Fig. 2. For this reason, we will additionally compare using S-IRLS-SGD and SQN as `opt()` in `update-mode()` in the real data example.

4.2. Example 2: simulated data

For this example, we performed full enumeration of all the 32 768 possible models using regular IRLS and S-IRLS-SGD with subsample sizes of 20%, 10%, 5%, 1%, 0.75%, 0.5%, 0.25%, 0.1% and 0.05% on both 10 000 and 100 000 observations and for both Gaussian and logistic models. Each experiment was repeated 20 times in order to create confidence bands for the estimates.

To evaluate the performance of the S-IRLS-SGD algorithm used as `opt()` in `update-mode()`, it is directly compared to regular IRLS. The parameters of interest are the *posterior marginal probabilities (PMPs)* of including the independent variables $p(\gamma_j = 1|\boldsymbol{y})$. S-IRLS-SGD was set to run 20 S-IRLS iterations for initialisation followed by 250 BSGD iterations for Gaussian models and 75 S-IRLS and 500 BSGD iterations for logistic models for all the experiments.

For each subsample size in S-IRLS-SGD, the absolute difference to the true value was calculated for the PMP of each γ_j. The same was also done using the best estimate for each model from 5, 10 and 20 runs of the algorithm. The mean of the 20 runs for 10 000 observations of logistic models, with 95% confidence intervals is presented in Fig. 3. Also, we illustrate lines for the estimates created from the 5, 10, and 20 runs by selecting the maximum estimate for each model across these runs. The most apparent tendency is that with the increasing size of the subsamples used in S-IRLS-SGD the estimates are getting closer to the true values. In this example, using less than 1% of the data gives already quite acceptable results. We also notice that when selecting the best estimate per model using a higher number of runs, the absolute difference to the true values gradually decreases.

Additional plots for 100 000 observations with logistic models, as well as 10 000, and 100 000 observations with Gaussian models are available in Figs. D.14, D.15 and D.16 in Appendix D. In these figures, we see similar trends.

4.3. Example 3: simulated data (MJMCMC)

In Example 3, we perform Bayesian variable selection with MJMCMC using Algorithm 3 with S-IRLS-SGD as `opt()` algorithm in `update-mode()`. We present results for the renormalised estimate (12), for the Markov chain estimate (9) and additionally for estimate (4) based on full enumeration where marginal likelihoods are estimated via subsampling. In these experiments, estimates of the PMP can be compared with the true values obtained from (4) using the full dataset. Following [14] and [25], we assess the performance of algorithms using the *root mean square error (RMSE)*, which for K independent runs of the MJMCMC can be calculated using

$$\text{RMSE}(\gamma_j) = \sqrt{\frac{\sum_{k=1}^{K}\left[\hat{p}_k(\gamma_j = 1|\boldsymbol{y}) - p(\gamma_j = 1|\boldsymbol{y})\right]^2}{K}}. \tag{31}$$

Here $\hat{p}_k(\gamma_j = 1|\boldsymbol{y})$ is the estimate obtained in the kth run. We will show empirically that under subsampling (using the same `update-mode()`) MJMCMC achieves *better* results than the full enumeration (full enumeration is also run with stochastic optimisation and with exactly the same sub-sample size and tuning parameters but a single visit of every model).

Using the data from Example 1, MJMCMC was run using both regular IRLS and S-IRLS-SGD with a subsample of size 5%, 1%, 0.75%, 0.5% and 0.25%, respectively. The algorithm was run 20 times for each subsample size and each run was allowed to perform 33 000 iterations. For the MC (11), RM (12) and full enumeration estimators, the marginal inclusion probabilities for the 15 covariates were estimated. These estimates were compared to the true values by calculation of the RMSE as defined in Equation (31). S-IRLS-SGD was set to run 20 S-IRLS and 250 BSGD iterations for Gaussian models and 75 S-IRLS and 500 BSGD iterations for logistic data for all the experiments.

To be able to assess how well the algorithm converges, the RMSE was calculated for the first 1 000, 2 000,..., 33 000 iterations. The results for the data with 100 000 observations are presented in Figs. 4 and 5. Additionally, Figs. E.17 and E.18 in Appendix E contain the results for the data with 10 000 observations. In all of the figures, we also present the RMSE for the full enumeration calculated using the same settings for S-IRLS-SGD.

Regarding the results for the Gaussian data, we see in Figs. 4 and E.17 that the MC estimates perform better than their RM counterparts when using subsampling. Further, the full enumeration performs *worse* than both estimates for all subsamples with a size smaller than 5%. At 5% the MC estimate outperforms full enumeration after around 5 000-10 000 iterations while the RM estimate roughly coincides with it from around 10 000. Convergence for the RM estimates appears to become very slow after the first 10 000 iterations. To some extent, this is also true for the MC estimates, although not as pronounced.

The results obtained for the logistic models show a very different pattern. In Figs. 5 and E.18, we see that for every subsample size, the RM estimates perform very well, converging very close to the true distribution after very few iterations. On the other hand, the MC estimates quickly attain an RMSE of between 0.1-0.15 after which the convergence becomes very slow. Here, however, it should be noted that the approximations of the marginal likelihoods are used, i.e. Laplace approximations do not yield exact results under Bernoulli observations. Looking at the results as a whole, we can, as expected, see that a larger subsample provides a better estimate, albeit at a higher computational complexity per iteration.

4.4. Example 4: real data (MAGIC gamma telescope dataset)

As a real data example, we will look at the `MAGIC Gamma Telescope Dataset` from the UCI database available at https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope. The dataset is concerned with the registration of high-energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using some kind of imaging technique. The response is a binary variable y of whether the telescope detects one of two classes $\{g, h\}$, standing for gamma (signal) or hadron (background). There are 10 explanatory variables in the dataset. All of them are numerical. The full sample

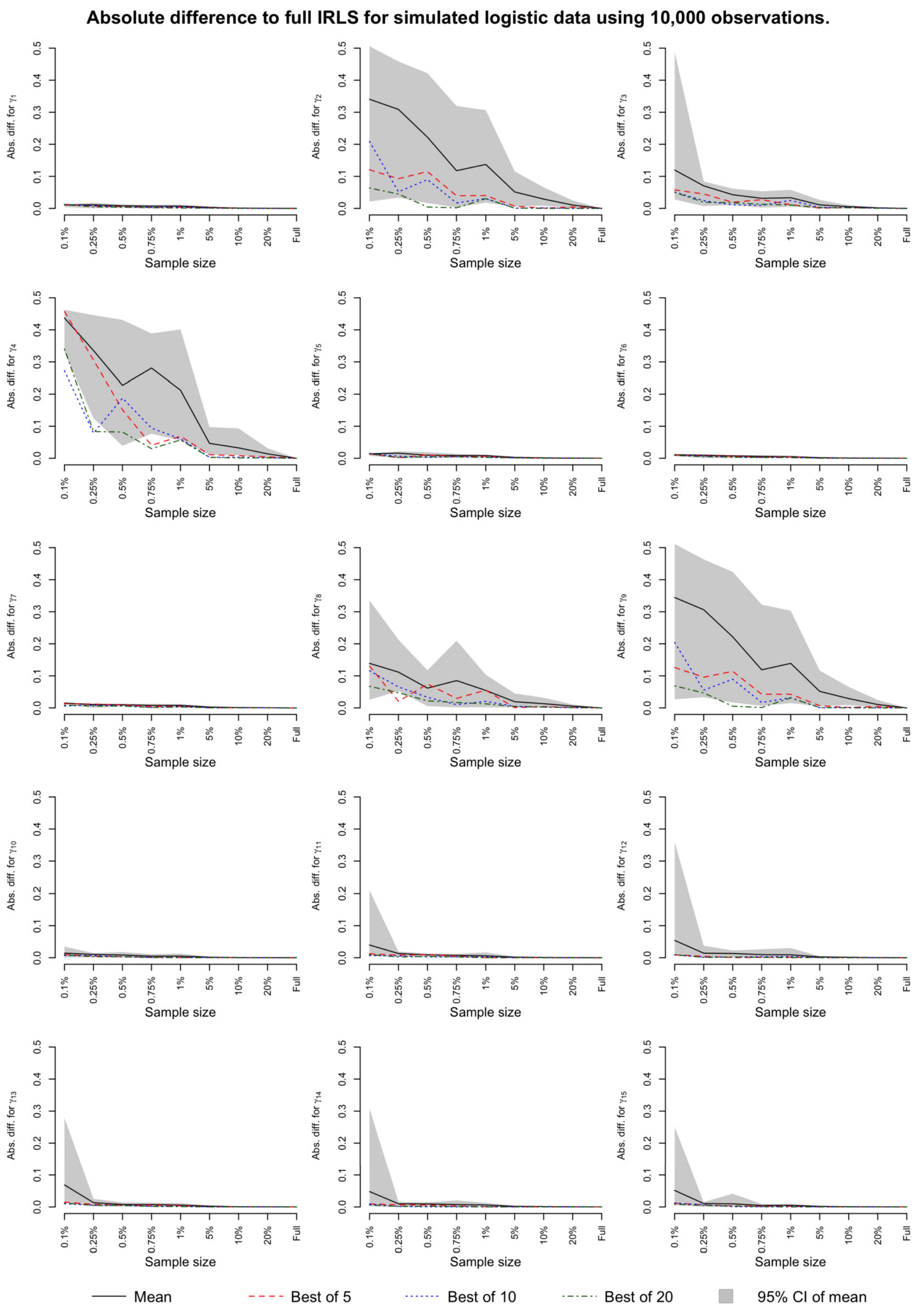

Fig. 3. The absolute difference for marginal posterior probabilities for full enumeration of 32 768 logistic models using 10 000 observations from the dataset in Example 2. 20 runs were performed and the black line shows the mean absolute difference, with grey 90% confidence intervals. The dashed and dotted lines show the absolute difference obtained by using the best likelihood estimate for each model that occurred during 5, 10 and 20 random runs.

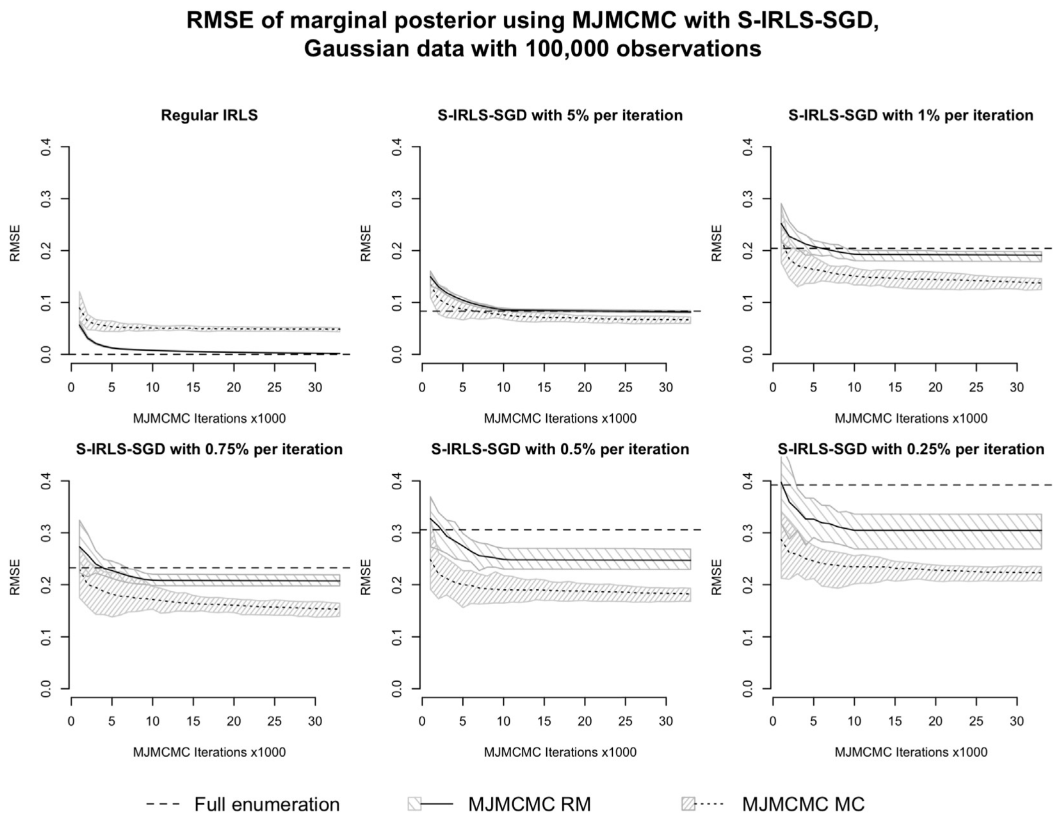

Fig. 4. Root mean squared error compared to the true distribution for MJMCMC using both regular IRLS and S-IRLS-SGD with varying subsample sizes. MJMCMC was run 20 times and for 33 000 iterations for every variant of the algorithm. The black solid and dotted lines show the mean RMSE for the 20 runs, with the shaded areas showing 90% confidence intervals, for RM and MC estimates respectively. The dashed line shows the RMSE for the full enumeration using the same algorithm settings.

size is $n = 19\,020$ with $n_g = 12\,332$ and $n_h = 6\,688$ corresponding to the number of signal and background observations respectively. The variables are as follows:

- $fLength$: major axis of ellipse [mm]
- $fWidth$: minor axis of ellipse [mm]
- $fSize$: 10-log of the sum of the content of all pixels [in photo]
- $fConc$: ratio of the sum of the two highest pixels over $fSize$ [ratio]
- $fConc1$: ratio of highest pixel over $fSize$ [ratio]
- $fAsym$: distance from the highest pixel to the centre, projected on the major axis [mm]
- $fM3Long$: 3rd root of the third moment along major axis [mm]
- $fM3Trans$: 3rd root of the third moment along minor axis [mm]
- $fAlpha$: angle of major axis with vector to origin [deg]
- $fDist$: distance from the origin to centre of ellipse [mm]

Further, we added squared values of each explanatory variable to make the search problem less trivial, yet still feasible for full enumeration to obtain the ground truth. Thus, we end up with 20 covariates and hence explore the model space of 2^{20} models. The same model specifications as in the simulation examples with binary responses were used.

To obtain the ground truth, a full enumeration of all models with IRLS optimisation was run. The corresponding marginal inclusion probabilities are depicted in Fig. 6. We see that 12 covariates have a marginal inclusion probability close to 1, 5 covariates - close to 0, and 3 somewhere in between.

Further, MJMCMC was run using regular IRLS, S-IRLS-SGD, and SQN algorithms. For the latter two, we used a subsample size of 1% per iteration. The MJMCMC algorithm was run 20 times for each `update-mode()` and each run was allowed to perform 15 000 iterations. Additionally, S-IRLS-SGD was set to run 75 S-IRLS and 500 BSGD iterations, whilst SQN was run for 2 500 iterations. Using both the MC (11) and RM (12) estimators, the marginal inclusion probabilities for the 20 covariates were estimated. Following our simulation examples, these estimates were compared to the true values by calculation of the RMSE as defined in Equation (31). The RMSE was calculated for the first 500, 1 000,..., 15 000 iterations.

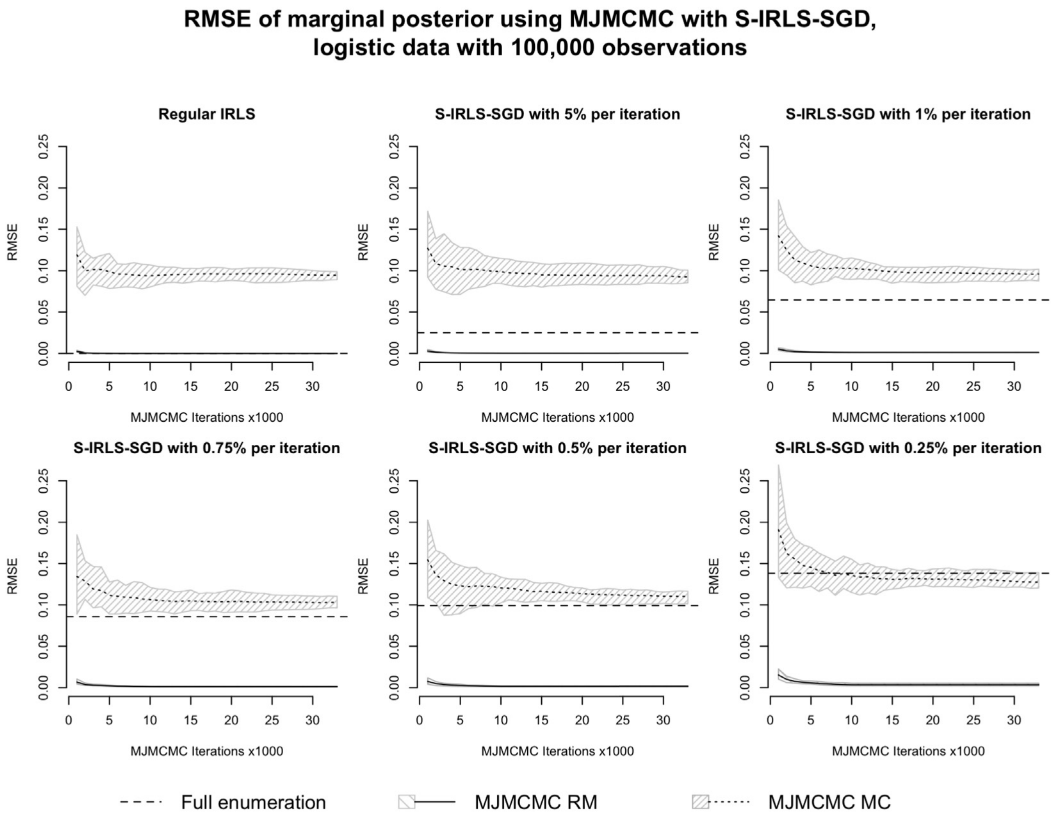

Fig. 5. See caption of Fig. 4. Note that here the RM estimates have a very small error, for which reason they almost coincide with the *x*-axis in the plots.

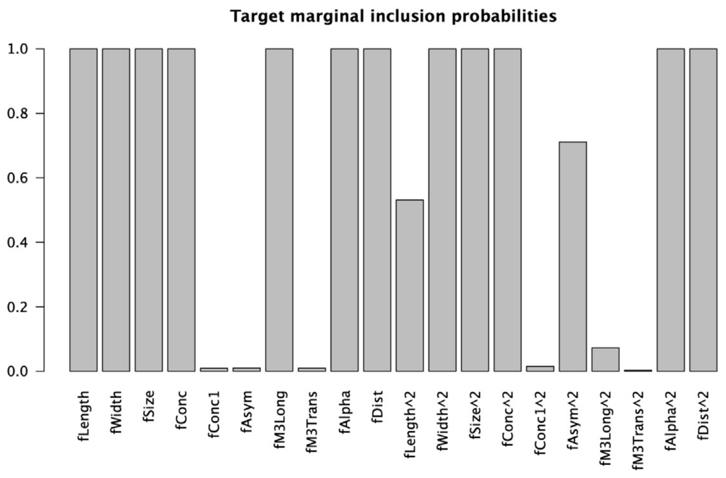

Fig. 6. Real target marginal inclusion probabilities were obtained under full enumeration and full IRLS optimisation.

The results are presented in Fig. 7. For MJMCMC combined with IRLS, we see that the RM estimates become indistinguishable from the truth in less than 2 500 MJMCMC iterations. Whilst MC estimates converge at a significantly slower pace. Further, just as we saw for the simulated data with Bernoulli distributed responses, under S-IRLS-SGD, the RM converge to the proximity of the true distribution after very few iterations, after which the convergence becomes very slow. For the same settings, the MC estimates perform significantly worse, although they continue to converge at a quicker pace even after 12 000 iterations. Also, both MC and RM estimates under S-IRLS-SGD `update-mode()` are significantly worse than those obtained under full IRLS. RM under S-IRLS-SGD is on par (somewhat better) than MC obtained under IRLS. Lastly, SQN `update-mode()` performed significantly worse in the sense of both RM and MC estimates than the two competing approaches. Interestingly, under SQN, the MC based estimates are somewhat better than the RM ones.

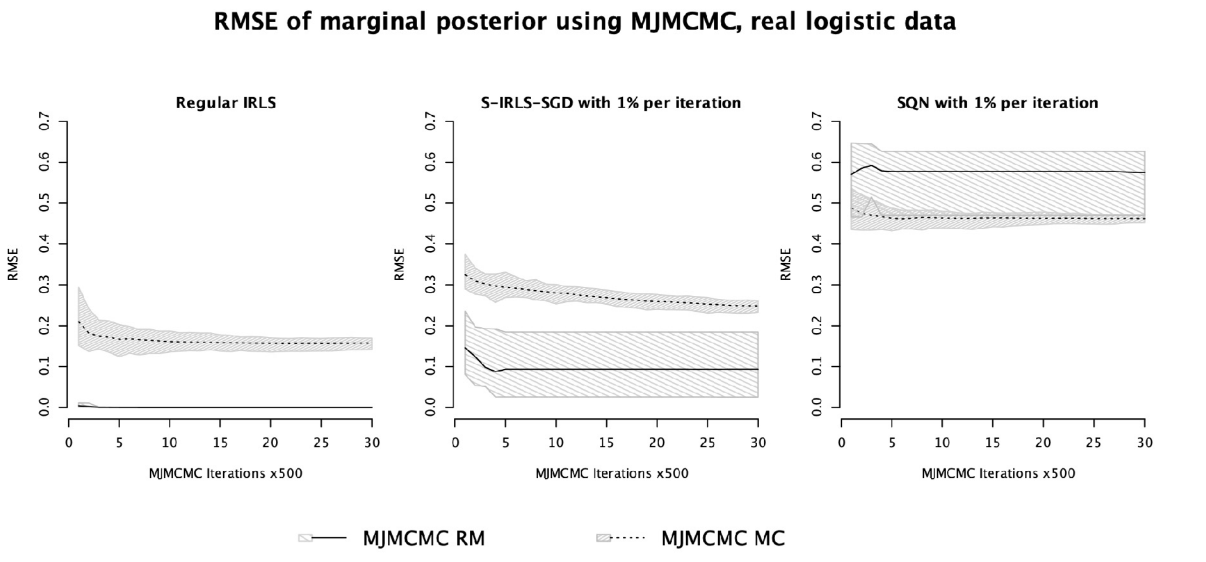

Fig. 7. Root mean squared error compared to the true distribution for MJMCMC using both regular IRLS, S-IRLS-SGD and SQN with a subsample size of 1% per iteration. MJMCMC was run 20 times and for 15 000 iterations for every algorithm. The black solid and dotted lines show the mean RMSE for the 20 runs, with the shaded areas showing 90% confidence intervals, for RM and MC estimates respectively.

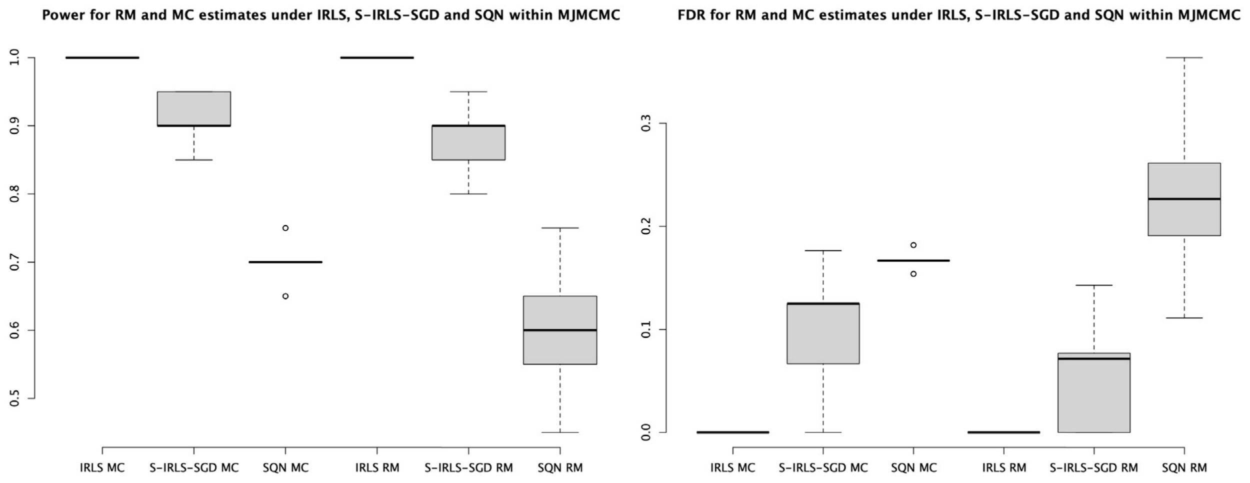

Fig. 8. Power and FDR for MJMCMC combined with IRLS, S-IRLS-SGD, and SQN optimisation as `opt()` in `update-mode()`.

It might be hard to interpret what the obtained overall RMSEs mean for the variable selection *in practice*. Hence, we are additionally evaluating the performance of different `update-mode()` optimisation algorithms for the model selection task. Variables are selected according to the median probability model [21], which is a standard approach that has attractive theoretical properties [4]. The median probability model considers a covariate selected if its marginal inclusion probability is above 0.5. We compared IRLS, S-IRLS-SGD, and SQN used for `update-mode()` under both MC and RM estimates of the marginal inclusion probabilities. Furthermore, for each of the 20 runs performed for each scenario, we evaluated statistical Power and false discovery rates (FDR) under the median probability selection rule. Here, the *truth* is assumed to correspond to selection according to the median probability rule after the full enumeration of all 2^{20} models. Box-plots of the results are summarised in Fig. 8 for Power and FDR. We observe that MJMCMC under full IRLS gives exactly the same results as full enumeration both for MC and RM estimates of the marginal inclusion probabilities. MJMCMC with S-IRLS-SGD used as `update-mode()`, reaches both for MC and RM estimates a Power of 0.9 on average with some variation across the runs. The corresponding FDR is slightly above 0.1 on average for MC estimates and slightly below 0.1 for RM estimates, again with some variation across the runs. These results show that the performance under S-IRLS-SGD `opt()` in `update-mode()` is practically very reasonable with a fraction of time used for optimisation when compared to the full IRLS. Again, the SQN `opt()` in `update-mode()` is performing significantly worse than other methods with an average Power of around 0.7 for MC estimates and 0.6 for the RM estimates. FDRs for MC and RM estimates under the SQN `opt()` in `update-mode()` are also the highest with the former slightly below 0.2 and the latter - slightly above 0.2 on average. In spite of the rather high RMSEs of the SQN approach its practical performance on the model selection task is still relatively reasonable, though by far not as good as the performance of S-IRLS-SGD.

To understand why model selection under SQN and S-IRLS-SGD `opt()`'s in `update-mode()` did not give perfect Power and FDR, we further depicted box-plots of the marginal inclusion probabilities of individual covariates for both MC and RM estimates. The results for MC and RM estimates respectively are presented in Figs. F.19 and F.20 in Appendix F of the paper. These results show that for some runs MC estimates of posterior inclusion probability under S-IRLS-SGD `opt()` in `update-mode()` are deviating from the reference values for $fConc1$, $fLength^2$, and $fConc1^2$. However, in general they are quite close to those obtained under the full IRLS. The same is true for the RM estimates, which had some false selection for $fAsymp^2$ and $fM3Long^2$ with a large variation across the runs for these two covariates. In some of the runs, perfect model selection takes place, whilst in other runs larger errors are observed. This is in line with the confidence bands of RMSE for RM estimates under S-IRLS-SGD depicted in Fig. 7. MJMCMC combined with SQN as the `opt()` in `update-mode()` results in MC estimates that are always off the median probability model for $fWidth$, $fConc1$, $fM3Trans^2$ and $fAlpha^2$ and sometimes off for $fDist$, $fLength^2$, $fWidth^2$, $fSize^2$, $fAsymp^2$, and $fDist^2$. The RM estimates under SQN are similar: They are always off the median probability model for $fWidth$, and $fConc1$, and $fAlpha^2$ and sometimes off for all other covariates except for $fLength$, $fSize$, $fM3Long$, $fAlpha$, and $fM3Long^2$, for which they are always correct.

We see that the accuracy under SQN appears to be substantially lower than that under IRLS and S-IRLS-SGD. SQN does not provide a better computational time than S-IRLS-SGD either. For that reason, we currently would not recommend using the SQN-based second-order stochastic optimisation in the `update-mode()`.

5. Discussion

In this paper, we presented a novel approach to combining a stochastic optimisation algorithm for computing the marginal likelihood and MCMC algorithms for Bayesian model selection through a time-inhomogeneous Markov chain Monte Carlo approach. We also present a practical stochastic optimisation algorithm called S-IRLS-SGD which is designed to work well with the novel approach. In Example 1, we demonstrated that S-IRLS-SGD is better suited for estimating parameters for numerous models in the model space than standard stochastic optimisation algorithms including BSGD, SQN, adaQN and oLBFGS. The S-IRLS-SGD algorithm has been implemented as a package in the R programming language with computationally intense blocks written in C++. It is publicly available at https://github.com/jonlachmann/irls.sgd. Also, SQN, adaQN and oLBFGS optimisation algorithms in R were improved by implementing computationally intense blocks in C++. The improved package is available on GitHub at https://github.com/jonlachmann/stochQN.

In Section 3, we presented a general TIMCMC algorithm and provided two theorems outlining possible strategies for combining stochastic optimisation of marginal likelihood algorithms and MCMC. By reestimating the models each time they are visited but always selecting the best estimated marginal likelihood considered so far, we show that both Monte Carlo and renormalised estimates will converge to the exact Laplace approximation of the marginal likelihood. For Gaussian observations and certain types of priors (e.g. g-priors), the approximation coincides with the exact marginal likelihood. Throughout the simulation examples, we have demonstrated how the results of Theorems 1 and 2 are confirmed in practice.

Firstly, to evaluate the performance of the S-IRLS-SGD algorithm to compute marginal likelihoods, we performed the experiments in Example 2. There, we compared the performance of S-IRLS-SGD with that of IRLS for obtaining a posterior distribution of the models by enumerating the whole model space. The results obtained show that we are able to get an acceptable level of precision given a rather small subsample size, encouraging us to continue on the chosen path. By selecting the highest estimate per model out of 5, 10 and 20 runs, we observed that more evaluations result in even better estimates. Thus, we were able to provide empirical support for Theorem 1, i.e. that multiple evaluations of a given model under S-IRLS-SGD provide convergence towards the true posterior distribution of the renormalised estimates of the posteriors of interest.

TIMCMC visits models of higher importance more often, which implies that the estimates for these models converge towards their true marginal likelihood faster than for any arbitrary model. This allowed us to hypothesise that under S-IRLS-SGD for computing marginal likelihood, TIMCMC would be able to provide *even better* results than full enumeration using the same approximate optimisation algorithm. To demonstrate this, in Example 3, we combined S-IRLS-SGD with an MJMCMC algorithm into a TIMCMC framework. The experiments were carried out using simulated data. We confirmed that MJMCMC in most cases is able to provide a better estimate of the posterior marginal inclusion probabilities than full enumeration with the same settings for S-IRLS-SGD. For the logistic models, the RMSE of the RM estimates is very low, meaning that we are very close to the true distribution, even when we were just using 0.25% of the data at each iteration. For Gaussian data, S-IRLS-SGD in MJMCMC performs significantly worse, probably owing to the smaller budget for iterations used. Yet, a convergence trend was still numerically confirmed.

It is important to note that in some cases, the empirical rate of convergence of the implemented approach can deteriorate drastically after the algorithm has run for some time. To illustrate this, we tried to run the TIMCMC algorithm for a significantly larger amount of iterations than in the presented examples. The results obtained for one run with one million iterations (with a subsample size of 0.75% of the data having 100 000 observations) using the Gaussian data and model from Example 2 are shown in Fig. 9. There, we see that the renormalised estimate hardly improves after 200 000 iterations, whereas the MC estimate continues to converge even at around one million iterations, though at an extremely slow rate.

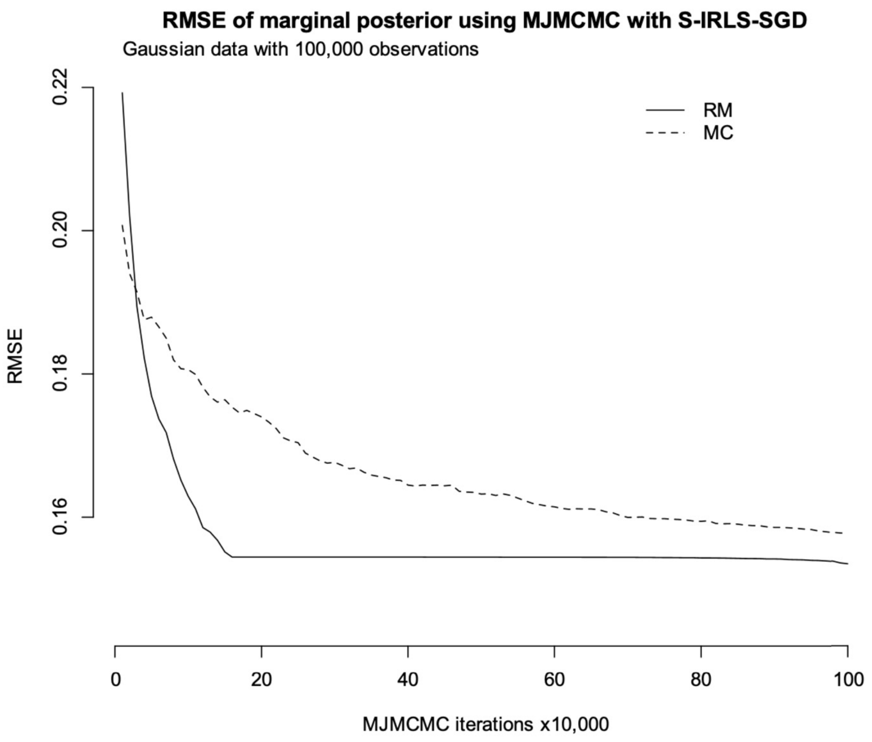

Fig. 9. Root mean squared error compared to the true distribution for MJMCMC using S-IRLS-SGD with a subsample size of 0.75%. MJMCMC was run for 1 000 000 iterations. The dashed lines show the RMSE for the MCMC estimates and the solid lines show RMSE for the renormalised ones.

Additional theoretical and experimental studies are required in the future to understand why convergence often becomes slower when the algorithm reaches a certain number of iterations.

Also, in Example 4, we provide a real data example, where we first confirm the conclusions about the convergence seen in Example 3. In addition, we include the SQN method as an alternative stochastic optimisation algorithm to be used in TIMCMC. We see that TIMCMC coupled with S-IRLS-SGD performs significantly better than SQN coupled with TIMCMC. Also, we have shown that despite some small RMSEs of the marginal posteriors of interest that are still present after a limited number of TIMCMC iterations (that we typically can afford running in practice), the Power and FDR for variable selection (which one is mainly interested in) are on the mark.

Even though S-IRLS-SGD used as a stochastic optimisation was shown to work well, it must be noted that there is extensive literature and knowledge about stochastic optimisation algorithms which could be applied here, and we are certain that there is a lot of room for possible improvements. More specifically, various extensions to SGD are available, possibly providing better global convergence properties for ill-conditioned likelihood functions as discussed by [46]. For example, modern variants of SGD optimisation like Adam, Adagrad, RMSprop or others from [8] could be used instead of the basic BSGD algorithm or S-IRLS-SGD. Also, novel stochastic second-order methods are available. We have tested SQN, adaQN, and oLBFGS, which did not work that well in practice, but there exist more recent advances in stochastic second-order methods available in the literature that might well improve the performance of TIMCMC further when used as `opt()` in `update-mode()`. More specifically, the LISSA method [1] seems promising to be used as our `opt()`. Yet, the focus of the paper was not to find the best performing stochastic optimisation method in combination with TIMCMC, but rather to give a principal novel way to combine a general stochastic optimisation and MCMC for computationally efficient Bayesian model selection through TIMCMC. We hope this framework will open new research directions in the future.

Lastly, as an alternative to the Laplace approximation for models with latent Gaussian structures, *Integrated Nested Laplace Approximation (INLA)* has emerged as an efficient approximation method [37]. In [24], it was shown empirically that INLA performs better than the Laplace approximation in various scenarios. It would, thus, be interesting to extend the sub-sampling ideas proposed in this article to that context by combining them with INLA. Another interesting idea for future research is to apply the proposed methods of [35] when doing the final calculation of the deviance. This would further improve the speed of the only step in the current algorithm that depends on the full dataset. Yet, the methods from [35] would break the regularity conditions of our Theorems 1 and 2, which means that additional theoretical studies would also be required to be able to incorporate it into our TIMCMC framework.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to Influence the work reported in this paper.

Appendix A. Optimization procedures

A.1. Gradient descent (GD)

The classical gradient descent algorithm was first proposed by [10], but also independently by [22]. It works on a function $f \in \mathcal{C}^1$ (continuously differentiable) by taking steps in the direction of the negative gradient calculated at the current point $\boldsymbol{\theta}$. If the objective is to be maximised, at a given point $\hat{\boldsymbol{\theta}}^{(t)}$, one step is defined as adding $\alpha\nabla f(\hat{\boldsymbol{\theta}}^{(t)})$ to the current point. For a step size α that is small enough, it holds that $f(\hat{\boldsymbol{\theta}}^{(t+1)}) \geq f(\hat{\boldsymbol{\theta}}^{(t)})$. This will lead to a monotonic sequence where $f(\hat{\boldsymbol{\theta}}^{(t)})$ is increasing, eventually reaching a maximum [8]. In our case, we are interested in maximising the log-posterior, i.e. $f(\boldsymbol{\theta}) = \ln p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})$, yet we shall continue this subsection with a general description of $f(\boldsymbol{\theta})$. We will further use $f_S(\boldsymbol{\theta})$ when f is evaluated on a subsample S, e.g. $f_S(\boldsymbol{\theta}) = \ln p(\boldsymbol{y}_S|\boldsymbol{x}_S, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})$

A.2. Stochastic gradient descent (SGD)

Originally proposed by [36], *stochastic gradient descent (SGD)* can be considered as a modification of regular gradient descent. Assume there are n observations. SGD works by replacing the gradient at each iteration with an approximation using only one data point, i.e. $\widehat{\nabla} f(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}[\ln p(y_i|x_i, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})]$ where i denotes one observation of the data chosen randomly at each iteration. The method also requires that the estimator of the gradient is unbiased, i.e. $E[\widehat{\nabla} f(\boldsymbol{\theta})] = \nabla f(\boldsymbol{\theta})$. The latter is easily satisfied when the data are assumed conditionally independent.

In contrast to using the full gradient, the unbiased approximation obtained will be very noisy and might for some steps cause the algorithm to make a move in an unfavourable direction. However, it is much less costly to compute, allowing for many more iterations to be performed inside the same computational budget. SGD has received a lot of recent attention in the machine learning community where it is the workhorse of neural networks among other techniques. An important reason for this is that it scales very well when the amount of data is large [8].

A.3. Batch stochastic gradient descent (BSGD)

A middle ground between GD and SGD is *Batch SGD (BSGD)*. Here the gradient is still estimated, but this is done using a subset or *batch* of data at each iteration, with size b chosen at the users' discretion. There exist variants that either go through the data in a fixed or random order. For the former, each batch consists of the b observations located immediately after the previous (starting at the beginning when all batches have been visited). The other alternative is to randomly sample b observations at each iteration. The iteration complexity when optimising the MLE is here $\mathcal{O}(bm)$, where m is the number of estimated parameters. Further, the convergence rate is generally between that of GD and SGD, owing to the more accurate (less noisy) approximation of the gradient [8].

A formal description of BSGD with new random samples at each iteration is provided in Algorithm 5. Here the step size is allowed to change with t. For $b = 1$, the algorithm reduces to SGD while for $b = n$ it is equal to the GD algorithm. Many implementations apply other stopping criteria. It is, for instance, possible to run until no improvements larger than a small number ε are made, or for a set number of iterations.

Algorithm 5: Gradient descent with convergence tolerance $\varepsilon > 0$ and step size α.

```
Initialise a first solution $\hat{\boldsymbol{\theta}}^{[0]}$.
while $|\hat{\boldsymbol{\theta}}_t - \hat{\boldsymbol{\theta}}_{t-1}| > \varepsilon$ do
    Draw a subsample $S$ of size $b$ without replacement from $\{1, \dots, n\}$.
    Calculate the gradient based on the subsample: $g_S = \nabla[\ln p(\boldsymbol{y}_S|\boldsymbol{x}_S; \hat{\boldsymbol{\theta}}^{(t)} + \ln p(\hat{\boldsymbol{\theta}}^{(t)})]$.
    Update $\hat{\boldsymbol{\theta}}_{t+1} = \hat{\boldsymbol{\theta}}_t + \alpha_t g_S$.
end
```

An essential requirement for the sequence $\{\hat{\boldsymbol{\theta}}^{(t)}\}$ is convergence to its mode (20), see Theorem 1. Such convergence guaranties are both based on some regularity conditions of the likelihood function $p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$ and on the shape of the step size sequence $\{\alpha_t\}$. For the generalised linear models considered here, these regularity conditions are satisfied, except for some degenerate cases like perfect separation, see [46] for details. If in addition, the step size sequence satisfies

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty,$$

then convergence in probability is guaranteed [8]. Criteria for convergence almost surely are considered in [33] where also more general (non-convex) settings are considered. More specifically, if we satisfy regularity conditions of Theorem 2 from [33], then $\hat{\boldsymbol{\theta}}^t_{\mathfrak{m}} \xrightarrow{a.s.} \hat{\boldsymbol{\theta}}_{\mathfrak{m}}$. Hence, by Theorem 1 $\hat{p}^{(t)}_{RM}(\mathfrak{m}|\boldsymbol{y}) \xrightarrow{a.s.} p_L(\mathfrak{m}|\boldsymbol{y})$ and by Theorem 2, $\hat{p}^{(t)}_{MC}(\mathfrak{m}|\boldsymbol{y}) \xrightarrow{a.s.} p_L(\mathfrak{m}|\boldsymbol{y})$.

A.4. Stochastic Quasi-Newton methods

For this family of methods, we compute not only the estimates of the gradient on a sub-sample but also those of the Hessian, which is then added into the update rule. This results in a general Algorithm 6 presented below.

Algorithm 6: Stochastic quasi-Newton with convergence tolerance $\varepsilon > 0$ and step size α.

```
Initialise a first solution $\hat{\boldsymbol{\theta}}^{[0]}$.
while $|\hat{\boldsymbol{\theta}}_t - \hat{\boldsymbol{\theta}}_{t-1}| > \varepsilon$ do
    Draw a subsample $S$ of size $b$ without replacement from $\{1, \dots, n\}$.
    Estimate the gradient based on the subsample: $g_S = \nabla[\ln p(\boldsymbol{y}_S|\boldsymbol{x}_S; \hat{\boldsymbol{\theta}}^{(t)} + \ln p(\hat{\boldsymbol{\theta}}^{(t)})]$.
    Estimate the Hessian $H_s$ based on the sub-sample
    Update $\hat{\boldsymbol{\theta}}_{t+1} = \hat{\boldsymbol{\theta}}_t + \alpha_t H_S g_S$.
end
```

The three specific versions we implemented for this paper (SQN, adaAN, and oLBFGS) are different in how we are approximating H_s based on the sub-sample resulting in different computation costs and convergence properties. For additional details, see [8].

Appendix B. Reproducing previous results for MJMCMC

To verify that the implementation of MJMCMC contributed to this paper works as intended, some previous results from [25] were reproduced and compared against the original implementation. Examples B1 and B2 compare the performance of the new implementation of MJMCMC with the previous one using real and simulated data, the first example corresponds to Example 1 in [25] and the second one corresponds to Example 2 of the same paper.

Example B1: U.S. crime data

The *U.S. Crime Dataset* compiled by [45] contains crime rates for 47 U.S. states along with 15 covariates. Following [25], we will test our implementation of MJMCMC against it to verify that the algorithm performs as it should. With one dependent variable and 15 independent, it is possible to create 2^{15} different models, including one with only an intercept. [25] followed [14] in using Zellner's g-prior [47] with $g = 47$, something we will also do. This gives us the exact marginal likelihood

$$p(\boldsymbol{y}|\mathfrak{m}) \propto (1+g)^{(n-p-1)/2}(1+g[1-R^2_{\mathfrak{m}}])^{-(n-1)/2}, \tag{B.1}$$

where $R^2_{\mathfrak{m}}$ is the coefficient of determination for the model $\mathfrak{m}$.

Since the total number of possible models is reasonably small, we began by enumerating all of them to calculate the true marginal distribution of the components of the $\mathfrak{m}$ vector. We then ran MJMCMC on the data 20 times for a number of iterations to visit approximately the same number of unique models as in the results we wanted to replicate. The reason for the approximation is that the previous implementation of MJMCMC is configured such that it runs until it encounters a fixed number of either unique or total models, whereas our implementation runs for a fixed number of iterations. For every parameter estimate, we calculated the RMSE for each indicator variable γ_j as defined in Equation (31).

The true values and the RMSE multiplied by 100 of the renormalised (RM) and MCMC (MC) estimates are presented in Table B.2, along with the corresponding values from [25]. The posterior mass captured by MJMCMC is also shown there. We can see that our results for the RM estimates are very similar to those previously reported, and the MC estimates are a bit better than the ones obtained in the original implementation. It should however be noted that the total number of models visited in our runs is a bit higher, something caused by the different ways the two implementations of the algorithm are tuned. For a comparison to competing algorithms such as BAS, MC^3 and RS, we refer to Table 3 of [25] and to [14].

To demonstrate the convergence of the algorithm, we also show the RMSE plotted against the number of iterations for both RM and MC estimates in Fig. B.10. We note here that the RM estimates converge quicker and more stably towards the true distribution for all of the components.

Example B2: simulated data for logistic regression (MJMCMC)

In [25], a dataset was created to be used for logistic regression where the model space is highly multi-modal, aiming to properly assess the convergence in this more complicated setting. The covariance structure is visualised in Fig. B.11. The data consists of 20 independent variables $\boldsymbol{x}$ and one dependent variable $\boldsymbol{y}$, each with 2 000 observations. This results in $2^{20} = 1,048,576$ possible models, providing a model space that is much larger and more computationally intensive to fully enumerate than in the previous example. This highlights the need for a good search algorithm to properly explore it without having to calculate every possible model. In this example, the number of observations is also larger. Hence, even the time it takes to calculate a single model becomes much higher than in the previous example.

We used the AIC prior from [12] to be able to compare the results with those of [25]. Using AIC, the marginal likelihood is approximated via the deviance $D(\boldsymbol{y})$ as

Table B.2
RMSE × 100 for RM and MC estimates of $\mathfrak{m}$ marginal inclusion probabilities based on 2 × 20 different runs of MJMCMC in Example B1. Also shown is the mean RMSE and total posterior mass captured. Previous results reported in [25] in parentheses.

	True value	RM	MC	RM	MC
γ_8	0.16	7.76 (6.57)	6.27 (10.68)	6.22 (5.11)	6.19 (10.29)
γ_{13}	0.16	7.42 (7.46)	6.32 (10.54)	6.04 (5.60)	5.96 (10.19)
γ_{14}	0.19	8.62 (8.30)	6.33 (12.43)	7.01 (6.30)	6.26 (12.33)
γ_{12}	0.22	7.96 (6.87)	8.71 (13.61)	6.56 (5.57)	6.25 (13.64)
γ_5	0.23	7.69 (6.30)	6.51 (13.45)	5.76 (4.59)	5.64 (13.65)
γ_9	0.23	9.28 (9.49)	7.18 (16.21)	7.08 (7.40)	6.87 (16.21)
γ_7	0.29	7.10 (4.37)	4.84 (13.63)	4.93 (3.45)	3.36 (12.73)
γ_4	0.30	5.66 (6.18)	8.75 (19.22)	3.58 (3.79)	6.27 (17.31)
γ_6	0.33	8.26 (8.61)	6.94 (19.71)	5.53 (6.14)	4.55 (19.49)
γ_1	0.34	9.40 (11.32)	7.05 (22.68)	7.10 (7.29)	6.17 (20.50)
γ_3	0.39	4.40 (3.95)	9.93 (11.13)	2.00 (2.38)	7.41 (6.99)
γ_2	0.57	3.77 (5.92)	13.97 (13.21)	2.93 (3.82)	11.00 (9.03)
γ_{11}	0.59	3.81 (3.57)	9.61 (13.49)	1.43 (2.37)	5.96 (15.94)
γ_{10}	0.77	6.44 (7.62)	13.32 (7.28)	5.23 (5.97)	10.94 (4.78)
γ_{15}	0.82	5.98 (9.23)	14.51 (4.45)	5.04 (6.89)	12.34 (5.85)
	Mean RMSE	6.90 (7.05)	8.68 (13.45)	5.10 (5.11)	7.01 (12.60)
	Total mass	0.44 (0.58)		0.58 (0.71)	
	Unique models	1900 (1908)		3219 (3237)	
	Total models	6200 (3276)		11200 (5936)	

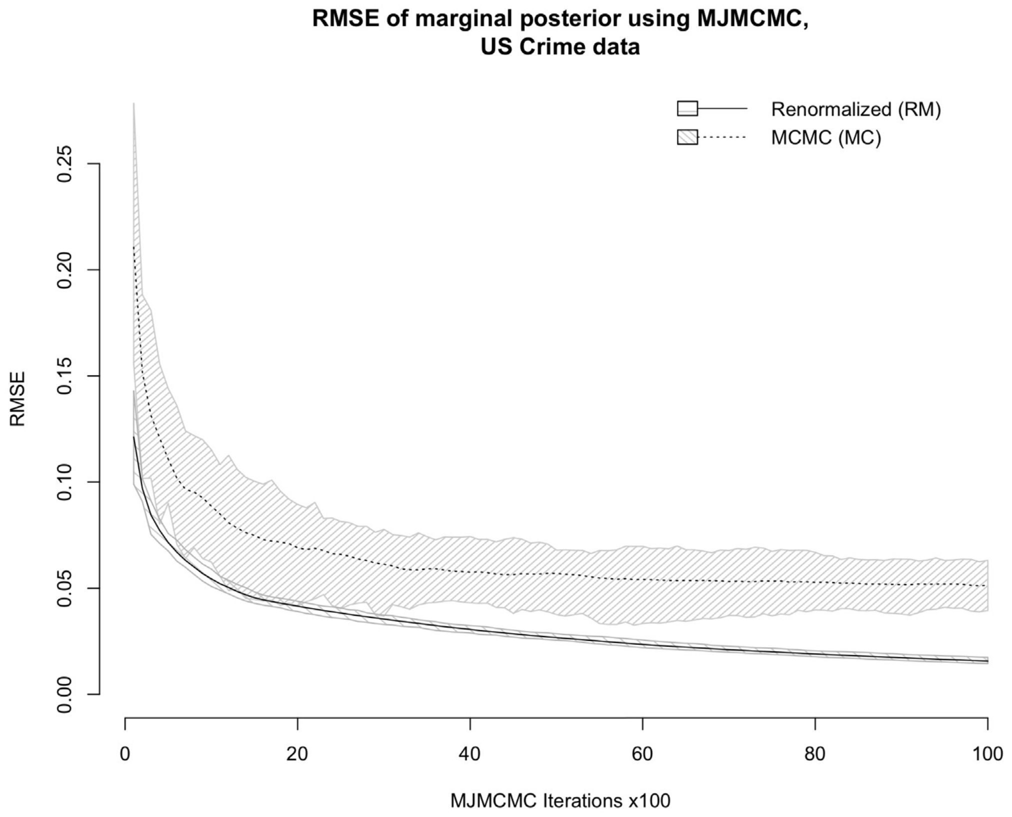

Fig. B.10. RMSE compared to the true distribution for MJMCMC run with the US Crime data in Example B1. MJMCMC was run 20 times and for 10 000 iterations. The black solid and dotted lines show the mean RMSE for the 20 runs, with the shaded areas showing 90% confidence intervals, for RM and MC estimates respectively.

$$\hat{p}(\boldsymbol{y}|\mathfrak{m}) \propto -\frac{1}{2}\left(D(\boldsymbol{y}) + 2\sum \gamma_i\right). \tag{B.2}$$

To provide a baseline to compare against we first performed a full enumeration of every possible model. This both gave us exact values and verified that our marginal likelihood was correctly implemented. We then ran MJMCMC 20 times for 32 768 iterations to be able to extract subsets of these runs that are comparable to the previous results. The RMSE after visiting a

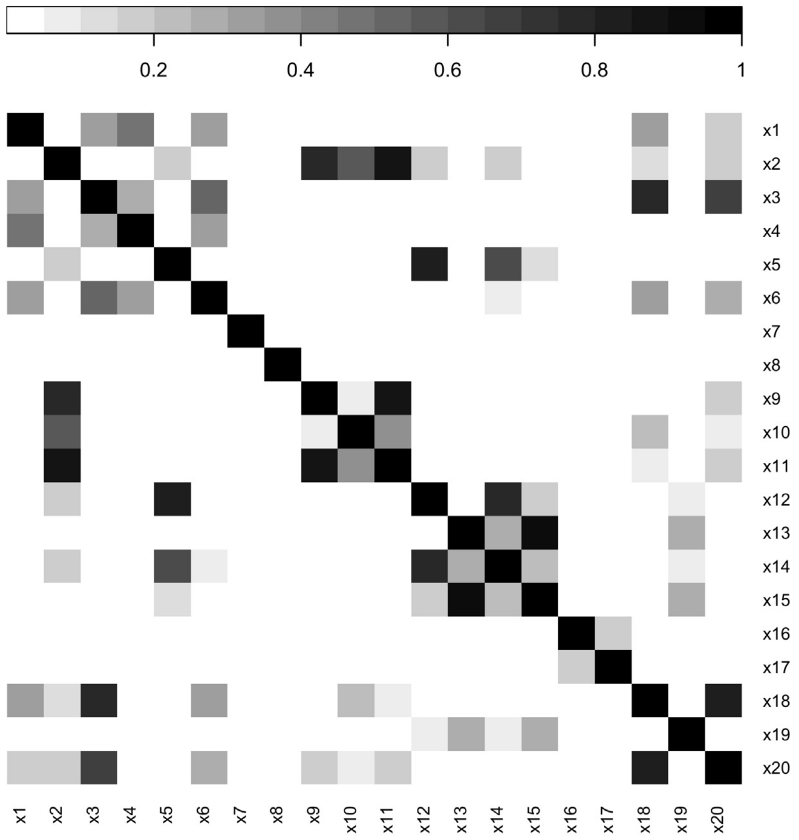

Fig. B.11. Absolute correlation between the exogenous x variables for the data used in Example 2. This complicated correlation structure ensures that the distribution of models is multi-modal.

Table B.3
Example B2. RMSE × 100 for RM and MC estimates of $\mathfrak{m}$ parameters based on 2 × 20 different runs of MJMCMC. Also shown is the mean RMSE and total posterior mass captured. Previous results reported in [25] in parentheses.

	True value	RM	MC	RM	MC
γ_6	0.29	7.96 (7.38)	7.20 (15.54)	5.56 (4.54)	7.20 (16.62)
γ_8	0.31	7.83 (6.23)	5.47 (15.50)	5.17 (3.96)	5.48 (16.94)
γ_{12}	0.35	4.92 (4.86)	8.99 (14.62)	3.19 (2.78)	8.37 (13.66)
γ_{15}	0.35	4.82 (4.55)	10.92 (15.24)	2.49 (2.56)	11.73 (15.45)
γ_2	0.36	5.15 (4.90)	6.88 (16.52)	3.45 (2.92)	7.08 (17.39)
γ_{20}	0.37	6.30 (4.82)	6.90 (14.35)	4.13 (2.66)	6.90 (14.08)
γ_3	0.40	6.89 (9.25)	12.20 (20.93)	4.23 (5.65)	12.87 (22.18)
γ_{14}	0.44	2.40 (3.14)	3.76 (17.54)	1.36 (1.58)	2.84 (16.24)
γ_{10}	0.44	2.82 (4.60)	6.35 (18.73)	1.09 (2.29)	6.13 (17.90)
γ_5	0.46	2.17 (3.10)	4.07 (17.17)	0.98 (1.53)	2.73 (16.97)
γ_9	0.61	4.02 (3.68)	3.76 (16.29)	2.51 (1.63)	3.07 (13.66)
γ_4	0.88	6.65 (5.66)	16.30 (6.70)	5.48 (3.74)	16.13 (6.26)
γ_{11}	0.91	3.97 (5.46)	14.27 (6.81)	2.92 (3.95)	14.13 (6.90)
γ_1	0.97	2.05 (1.90)	13.46 (1.74)	1.70 (1.35)	13.21 (1.34)
γ_{13}	1.00	0.00 (0.00)	11.31 (0.43)	0.00 (0.00)	11.13 (0.32)
γ_7	1.00	0.00 (0.00)	6.88 (0.57)	0.00 (0.00)	6.79 (0.41)
γ_{16}	1.00	0.00 (0.00)	6.37 (0.41)	0.00 (0.00)	5.99 (0.33)
γ_{17}	1.00	0.00 (0.00)	6.25 (0.43)	0.00 (0.00)	6.11 (0.39)
γ_{18}	1.00	0.00 (0.00)	6.31 (0.47)	0.00 (0.00)	6.19 (0.35)
γ_{19}	1.00	0.00 (0.00)	6.82 (0.52)	0.00 (0.00)	6.78 (0.36)
	Mean RMSE	3.40 (3.48)	8.22 (10.03)	2.21 (2.06)	8.04 (9.89)
	Total mass	0.54 (0.72)		0.73 (0.85)	
	Unique models	5151 (5148)		9993 (9988)	
	Total models	13250 (9998)		27200 (19849)	

similar number of unique models are presented in Table B.3, along with the true distribution and a comparison to previous results. In Fig. B.12, we also demonstrate the convergence of the RM and MC estimates over the course of 20 000 iterations.

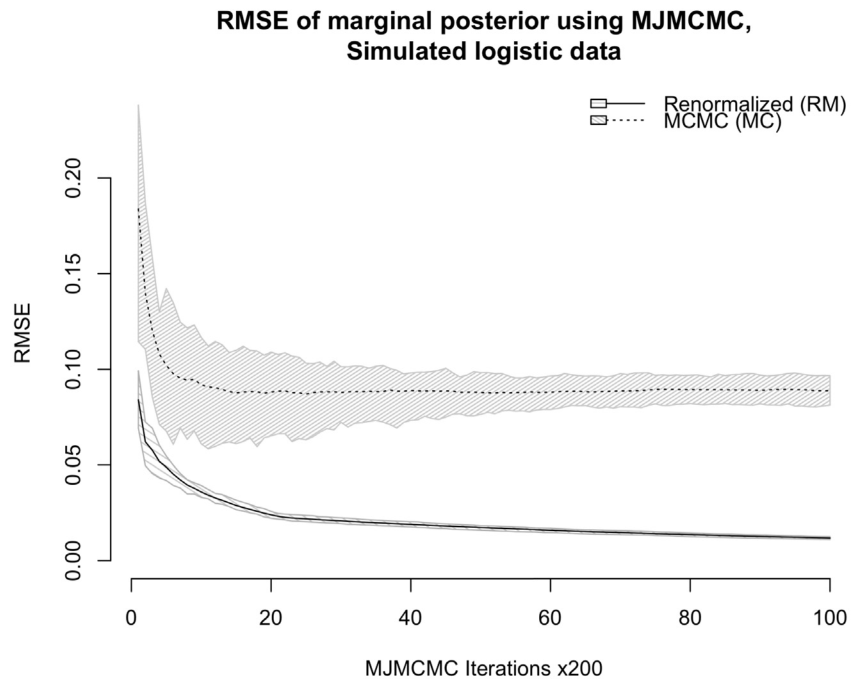

Fig. B.12. RMSE compared to the true distribution for MJMCMC run with the simulated logistic data in Example B2. MJMCMC was run 20 times and here the first 20 000 iterations are used to demonstrate the convergence. The black solid and dotted lines show the mean RMSE for the 20 runs, with the shaded areas showing 90% confidence intervals, for RM and MC estimates respectively.

By examining the results we see that the RM estimates are very similar to those of [25]. Just as in the previous example, the MC estimates are slightly more accurate in our new implementation. Also here it should be noted that while we estimated roughly the same number of unique models, we visited more models in total. Yet, we captured a bit less of the posterior mass than in the original implementation of MJMCMC.

Appendix C. Additional results for Example 1

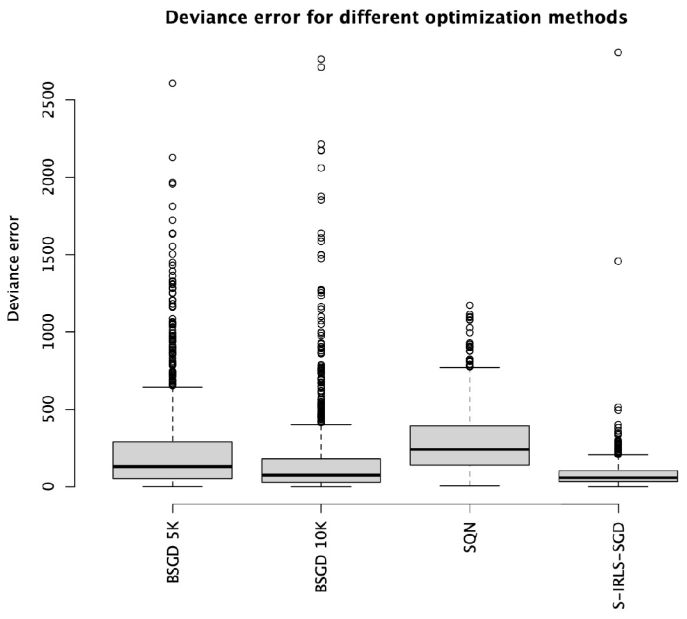

Fig. C.13. Boxplots of the errors of estimated deviances for the four best optimisation algorithms and tuning parameters from Table 1 and Fig. 2.

Appendix D. Additional results for Example 2

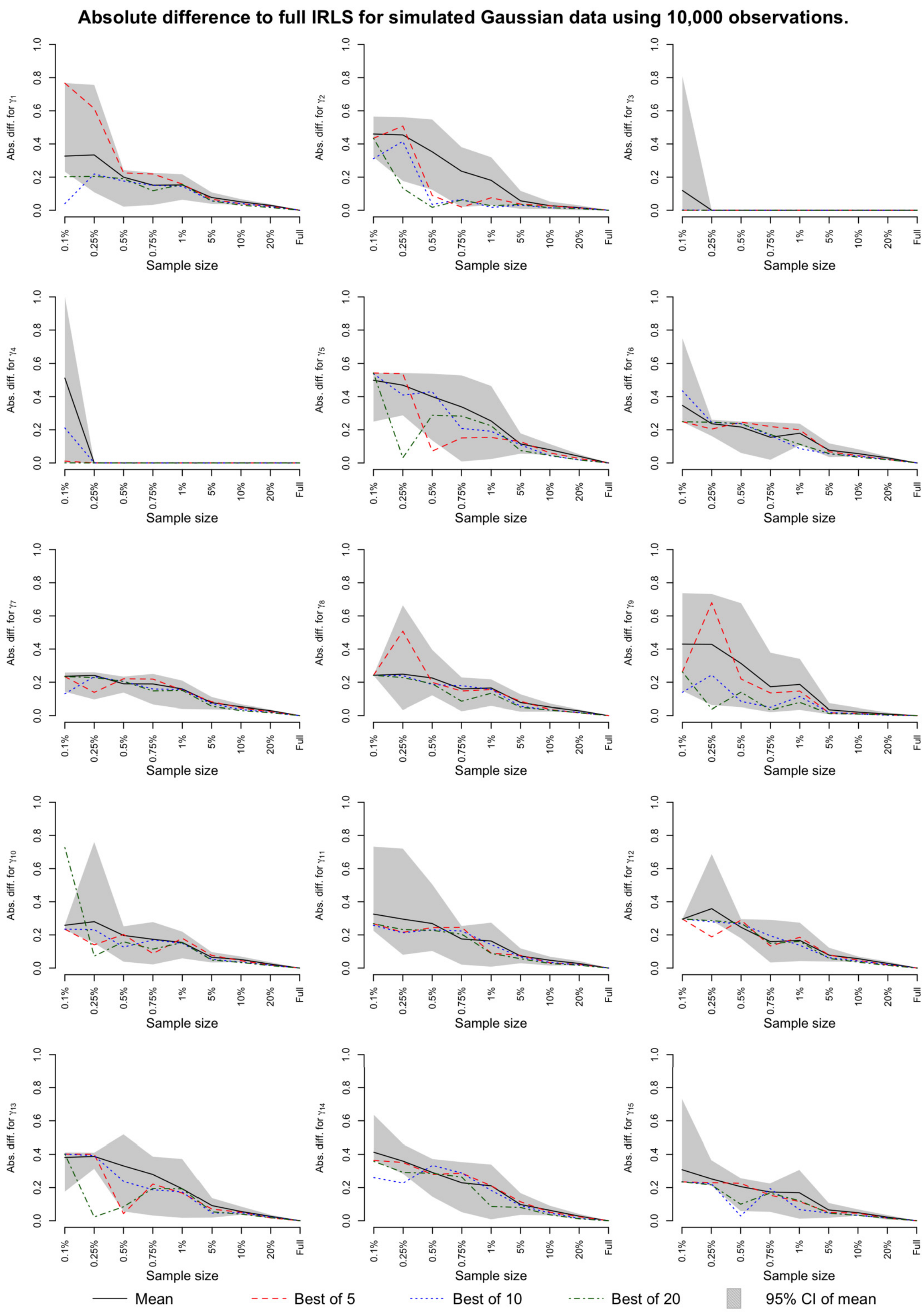

Fig. D.14. The absolute difference for marginal posterior probabilities for full enumeration of 32 768 Gaussian models using 10 000 observations from the dataset in Example 2. 20 runs were performed and the black line shows the mean absolute difference, with grey 90% confidence intervals. The dashed and dotted lines show the absolute difference obtained by using the best likelihood estimate for each model that occurred during 5, 10 and 20 random runs.

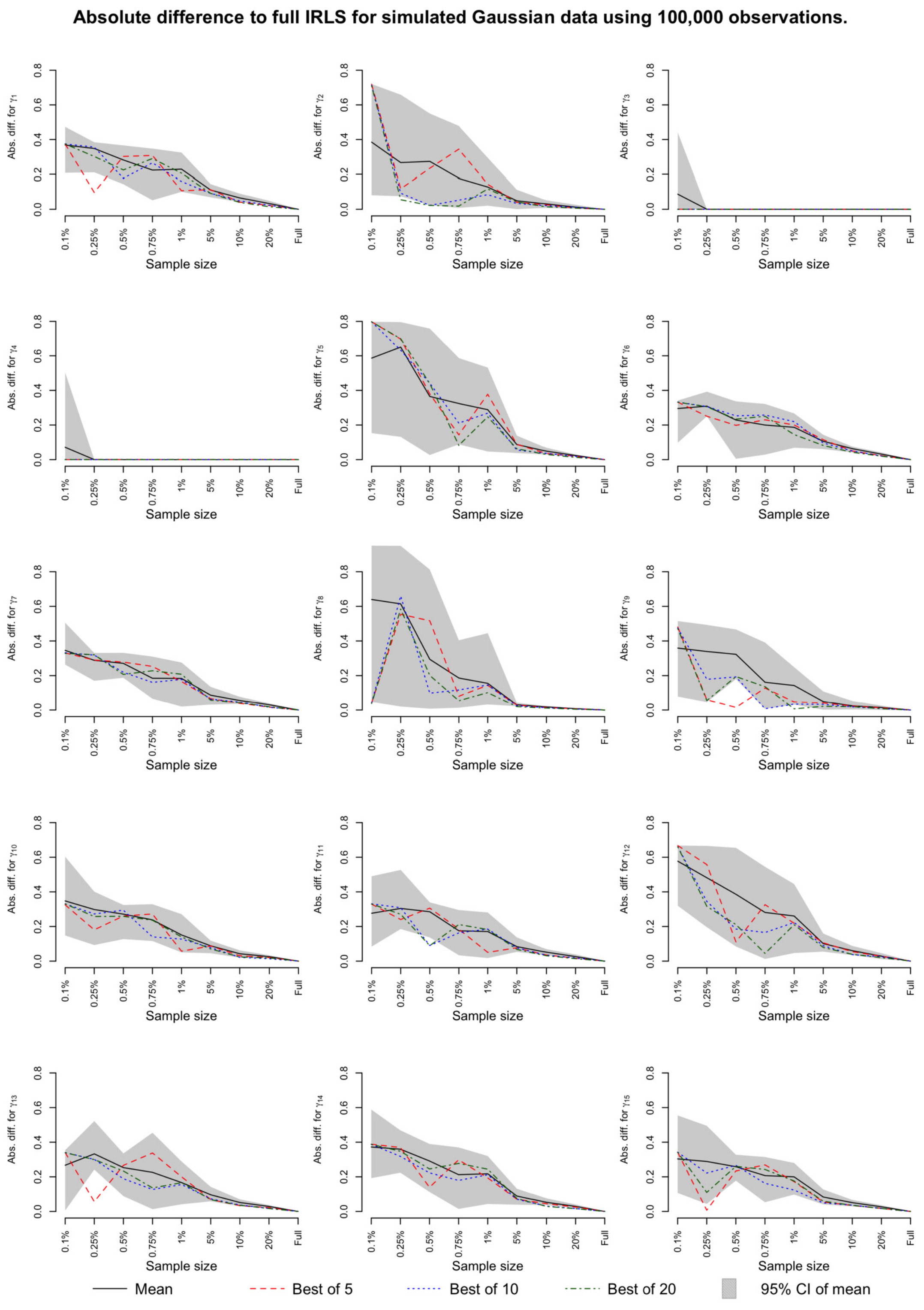

Fig. D.15. The absolute difference for marginal posterior probabilities for full enumeration of 32 768 Gaussian models using 100 000 observations from the dataset in Example 2. 20 runs were performed and the black line shows the mean absolute difference, with grey 90% confidence intervals. The dashed and dotted lines show the absolute difference obtained by using the best likelihood estimate for each model that occurred during 5, 10 and 20 random runs.

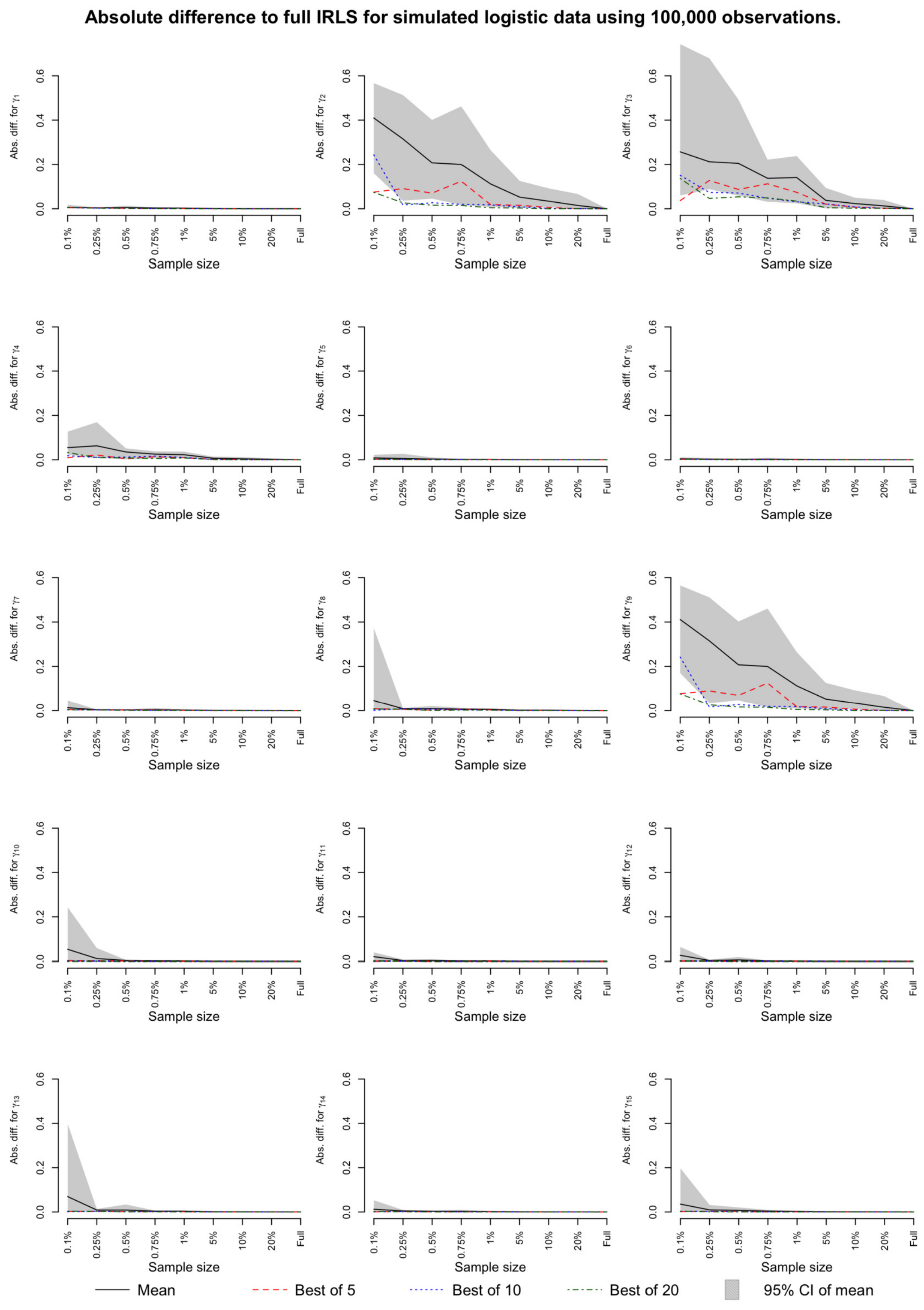

Fig. D.16. The absolute difference for marginal posterior probabilities for full enumeration of 32 768 logistic models using 100 000 observations from the dataset in Example 2. 20 runs were performed and the black line shows the mean absolute difference, with grey 90% confidence intervals. The dashed and dotted lines show the absolute difference obtained by using the best likelihood estimate for each model that occurred during 5, 10 and 20 random runs.

Appendix E. Additional results for Example 3

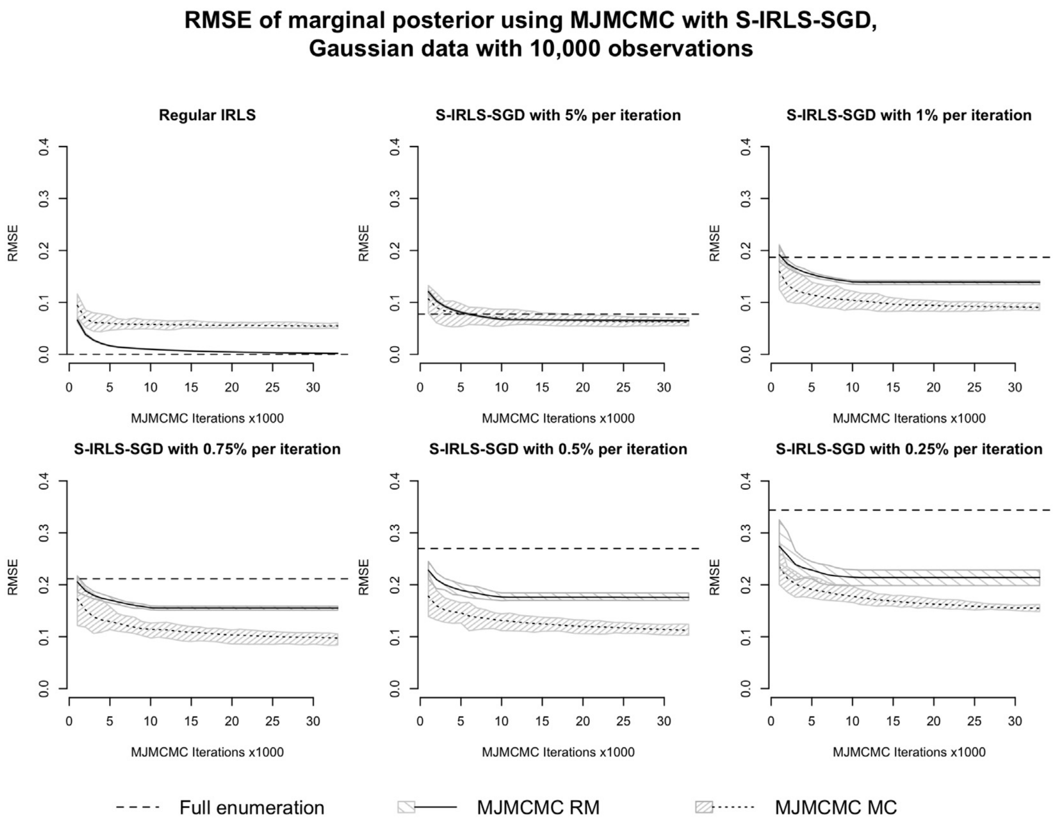

Fig. E.17. Root mean squared error compared to the true distribution for MJMCMC using both regular IRLS and S-IRLS-SGD with varying subsample sizes. MJMCMC was run 20 times and for 33 000 iterations for every variant of the algorithm. The black solid and dotted lines show the mean RMSE for the 20 runs, with the shaded areas showing 90% confidence intervals, for RM and MC estimates respectively. The dashed line shows the RMSE for the full enumeration using the same algorithm settings.

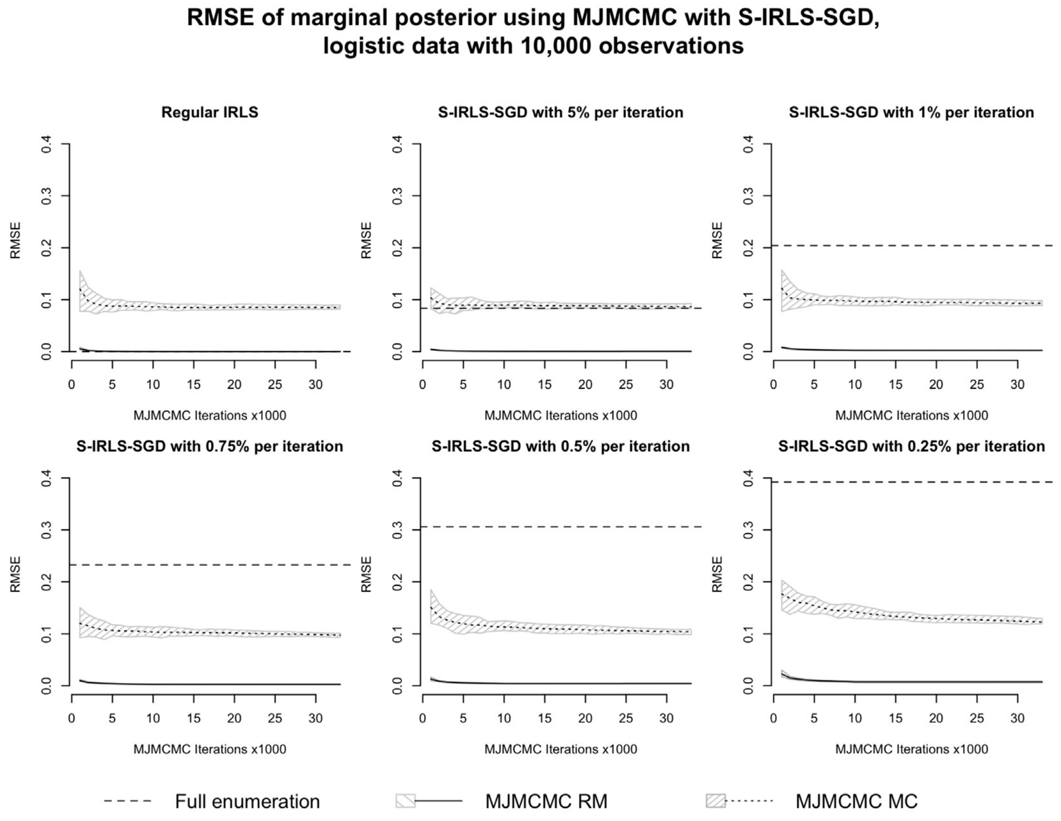

Fig. E.18. Root mean squared error compared to the true distribution for MJMCMC using both regular IRLS and S-IRLS-SGD with varying subsample sizes. MJMCMC was run 20 times and for 33 000 iterations for every variant of the algorithm. The black solid and dotted lines show the mean RMSE for the 20 runs, with the shaded areas showing 90% confidence intervals, for RM and MC estimates respectively. The dashed line shows the RMSE for the full enumeration using the same algorithm settings.

Appendix F. Additional results for Example 4

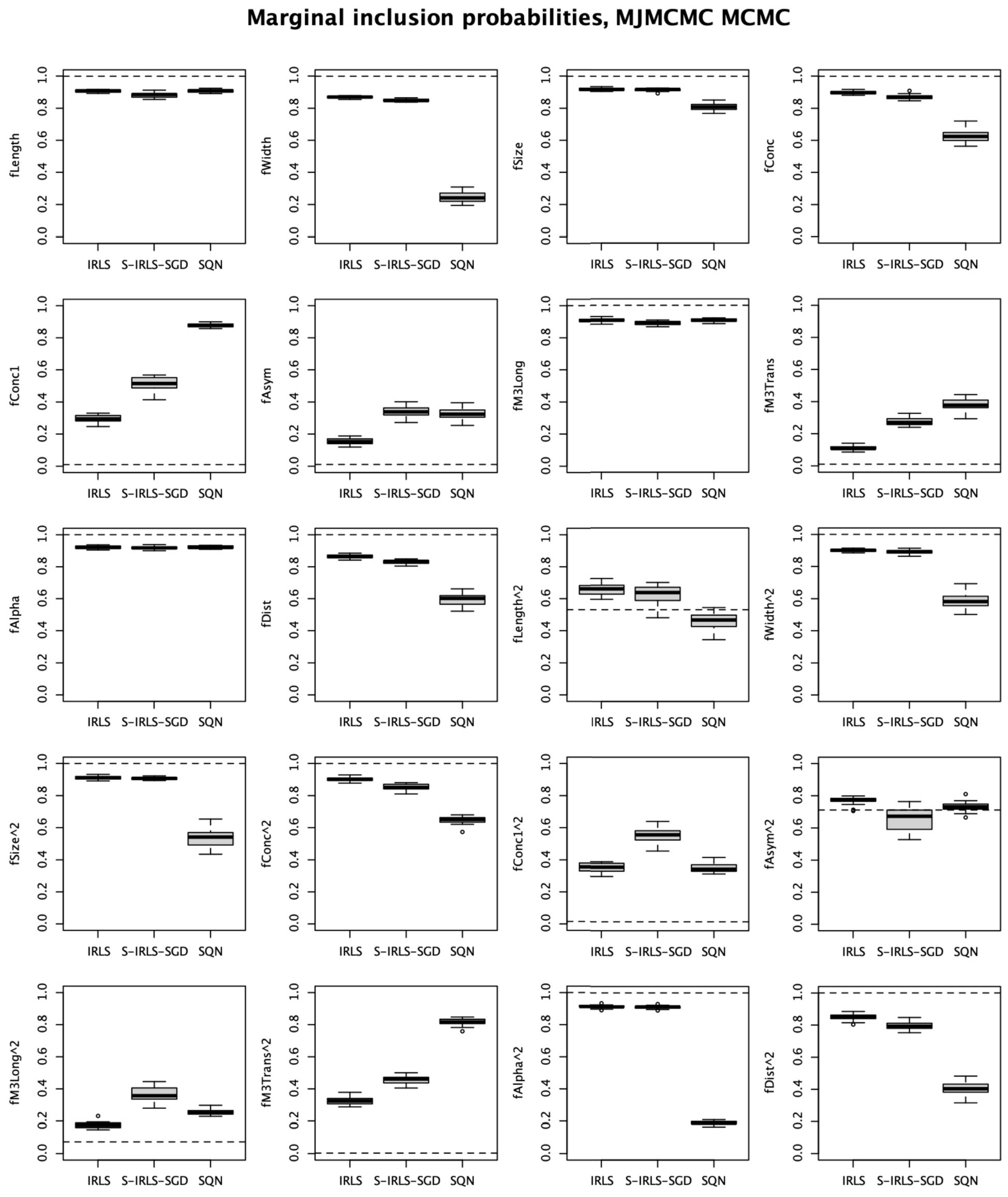

Fig. F.19. Marginal inclusion probabilities for the individual covariates in Example 4. The true probabilities are shown as dashed lines, and the MCMC estimates from MJMCMC using both regular IRLS, S-IRLS-SGD and SQN with a subsample size of 1% per iteration. MJMCMC was run 20 times and for 15 000 iterations for every algorithm.

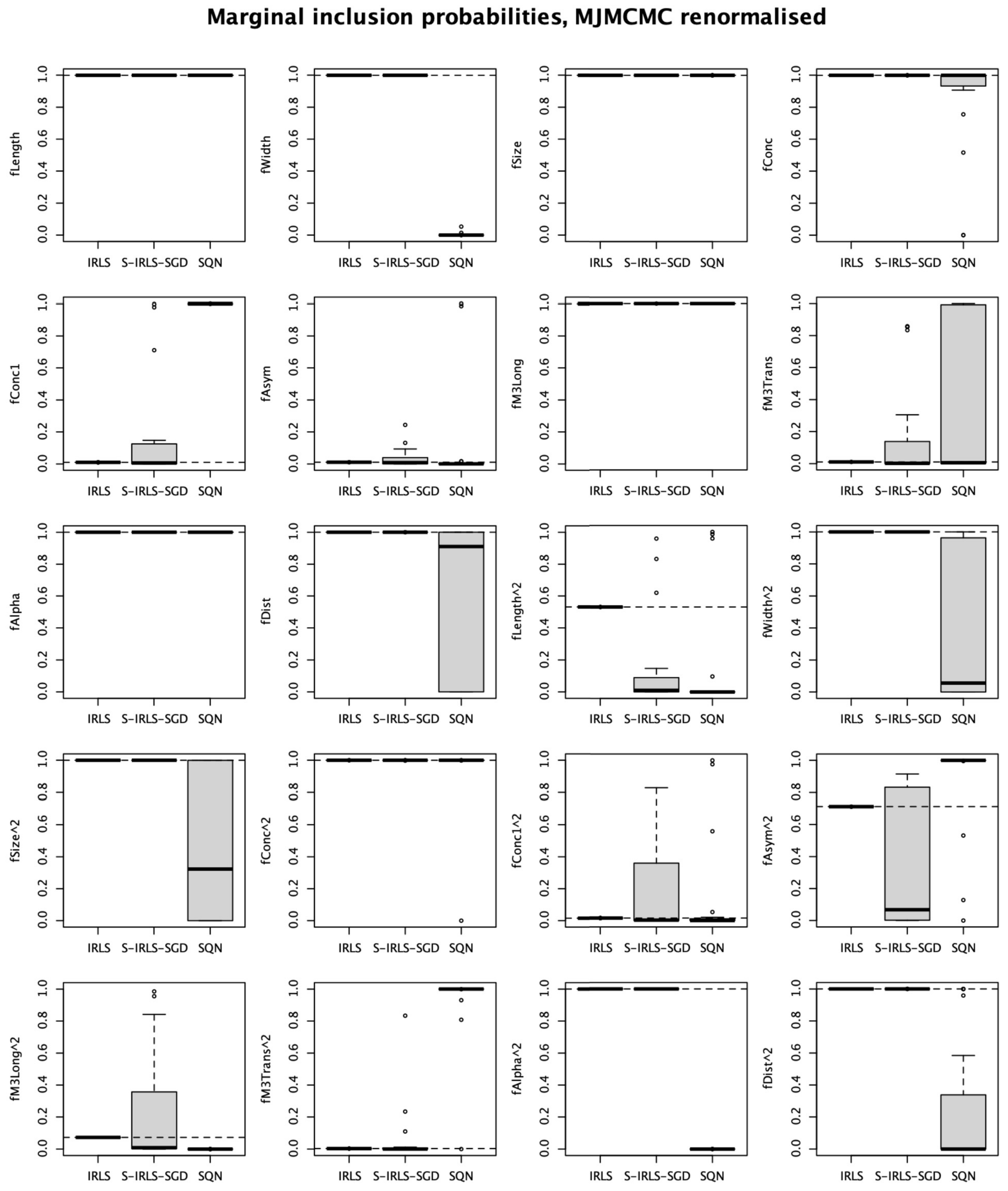

Fig. F.20. Marginal inclusion probabilities for the individual covariates in Example 2. The true probabilities are shown as dashed lines, and the renormalised estimates from MJMCMC using both regular IRLS, S-IRLS-SGD and SQN with a subsample size of 1% per iteration. MJMCMC was run 20 times and for 15 000 iterations for every algorithm.

References

[1] N. Agarwal, B. Bullins, E. Hazan, Second-order stochastic optimization for machine learning in linear time, J. Mach. Learn. Res. 18 (1) (2017) 4148–4187.
[2] T. Ando, Bayesian Model Selection and Statistical Modeling. Statistics: A Series of Textbooks and Monographs, Taylor & Francis, 2010.
[3] C. Andrieu, G.O. Roberts, The pseudo-marginal approach for efficient Monte Carlo computations, Ann. Stat. 37 (2) (2009) 697–725.
[4] M.M. Barbieri, J.O. Berger, Optimal predictive model selection, Ann. Stat. 32 (3) (2004) 870–897.
[5] A.E. Beaton, J.W. Tukey, The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data, Technometrics 16 (2) (1974) 147–185.
[6] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[7] L. Bottou, O. Bousquet, The tradeoffs of large scale learning, in: J. Platt, D. Koller, Y. Singer, S. Roweis (Eds.), Advances in Neural Information Processing Systems, vol. 20, Curran Associates, Inc., 2008.
[8] L. Bottou, F.E. Curtis, J. Nocedal, Optimization Methods for Large-Scale Machine Learning, 2018.
[9] R.H. Byrd, S.L. Hansen, J. Nocedal, Y. Singer, A stochastic quasi-Newton method for large-scale optimization, SIAM J. Optim. 26 (2) (2016) 1008–1031.
[10] A.-L. Cauchy, Methode generale pour la resolution des systemes d'equations simultanees, C. R. Math. Acad. Sci. Paris 25 (1847) 536–538.
[11] S. Chib, Marginal likelihood from the Gibbs output, J. Am. Stat. Assoc. 90 (432) (1995) 1313–1321.
[12] M. Clyde, BAS: Bayesian Variable Selection and Model Averaging using Bayesian Adaptive Sampling, 2020, R package version 1.5.5.
[13] M. Clyde, H. Desimone, G. Parmigiani, Prediction via orthogonalized model mixing, J. Am. Stat. Assoc. 91 (435) (1996) 1197–1208.
[14] M.A. Clyde, J. Ghosh, M.L. Littman, Bayesian adaptive sampling for variable selection and model averaging, J. Comput. Graph. Stat. 20 (1) (2011) 80–101.
[15] D. Cortes, stochQN: Stochastic Limited Memory Quasi-Newton Optimizers, 2019, R package version 0.1.2.
[16] R. Douc, E. Moulines, J.S. Rosenthal, Quantitative bounds on convergence of time-inhomogeneous Markov chains, Ann. Appl. Probab. 14 (4) (2004) 1643–1665.
[17] G. Fort, E. Moulines, P. Priouret, Convergence of adaptive and interacting Markov chain Monte Carlo algorithms, Ann. Stat. 39 (6) (2011) 3262–3289.
[18] N. Friel, J. Wyse, 2011, Estimating the evidence – a review.
[19] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-6 (6) (1984) 721–741.
[20] E.I. George, R.E. McCulloch, Approaches for Bayesian variable selection, Stat. Sin. 7 (2) (1997) 339–373.
[21] J. Ghosh, Bayesian model selection using the median probability model, Wiley Interdiscip. Rev.: Comput. Stat. 7 (3) (2015) 185–193.
[22] J. Hadamard, Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées, vol. 33, Imprimerie Nationale, 1908.
[23] A. Hubin, G. Storvik, Efficient mode jumping MCMC for Bayesian variable selection in GLMM, arXiv preprint, arXiv:1604.06398v3, 2016.
[24] A. Hubin, G. Storvik, Estimating the marginal likelihood with integrated nested Laplace approximation (INLA), arXiv preprint, arXiv:1611.01450, 2016.
[25] A. Hubin, G. Storvik, Mode jumping MCMC for Bayesian variable selection in GLMM, Comput. Stat. Data Anal. 127 (2018) 281–297.
[26] A. Hubin, G. Storvik, F. Frommlet, A novel algorithmic approach to Bayesian logic regression (with discussion), Bayesian Anal. 15 (1) (2020) 263–333.
[27] R.E. Kass, A.E. Raftery, Bayes factors, J. Am. Stat. Assoc. 90 (430) (1995) 773–795.
[28] N.S. Keskar, A.S. Berahas, AdaQN: an adaptive quasi-Newton algorithm for training RNNs, in: P. Frasconi, N. Landwehr, G. Manco, J. Vreeken (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, Cham, 2016, pp. 1–16.
[29] P.S. Laplace, Memoir on the probability of the causes of events, Stat. Sci. 1 (3) (1986) 364–378.
[30] H.B. Mann, A. Wald, On stochastic limit and order relationships, Ann. Math. Stat. 14 (3) (1943) 217–226.
[31] J. Matyas, et al., Random optimization, Autom. Remote Control 26 (2) (1965) 246–253.
[32] S.Y. Meng, S. Vaswani, I.H. Laradji, M. Schmidt, S. Lacoste-Julien, Fast and furious convergence: stochastic second order methods under interpolation, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 1375–1386.
[33] P. Mertikopoulos, N. Hallak, A. Kavis, V. Cevher, On the almost sure convergence of stochastic gradient descent in non-convex problems, Adv. Neural Inf. Process. Syst. 33 (2020) 1117–1128.
[34] M.A. Newton, A.E. Raftery, Approximate Bayesian inference with the weighted likelihood bootstrap, J. R. Stat. Soc., Ser. B, Methodol. 56 (1) (1994) 3–48.
[35] M. Quiroz, M. Villani, R. Kohn, M.-N. Tran, K.-D. Dang, Subsampling MCMC - an introduction for the survey statistician, Sankhya A 80 (1) (2018) 33–69.
[36] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. 22 (3) (1951) 400–407.
[37] H. Rue, S. Martino, N. Chopin, Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations, J. R. Stat. Soc., Ser. B, Stat. Methodol. 71 (2) (2009) 319–392.
[38] L. Saloff-Coste, J. Zúñiga, Merging and stability for time inhomogeneous finite Markov chains, Surv. Stoch. Proc. 4 (2011) 127.
[39] E.J. Schlossmacher, An iterative technique for absolute deviations curve fitting, J. Am. Stat. Assoc. 68 (344) (1973) 857–859.
[40] N.N. Schraudolph, J. Yu, S. Günter, A stochastic quasi-Newton method for online convex optimization, in: M. Meila, X. Shen (Eds.), Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, San Juan, Puerto Rico, in: Proceedings of Machine Learning Research, vol. 2, PMLR, 2007, pp. 436–443.
[41] G. Schwarz, Estimating the dimension of a model, Ann. Stat. 6 (2) (1978) 461–464.
[42] F.J. Solis, R.J.-B. Wets, Minimization by random search techniques, Math. Oper. Res. 6 (1) (1981) 19–30.
[43] L. Tierney, J.B. Kadane, Accurate approximations for posterior moments and marginal densities, J. Am. Stat. Assoc. 81 (393) (1986) 82–86.
[44] H. Tjelmeland, B.K. Hegstad, Mode jumping proposals in MCMC, Scand. J. Stat. 28 (1) (2001) 205–223.
[45] W. Vandaele, Participation in Illegitimate Activities, 1992, Ehrlich Revisited, 1960.
[46] R.W.M. Wedderburn, On the existence and uniqueness of the maximum likelihood estimates for certain generalized linear models, Biometrika 63 (1) (1976) 27–32.
[47] A. Zellner, P.K. Goel, Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti, North-Holland, 1986.