Properties of the Stochastic Approximation EM Algorithm with Mini-batch Sampling

Estelle Kuhn, Catherine Matias, Tabea Rebafka

manuscript No.
(will be inserted by the editor)

Properties of the Stochastic Approximation EM Algorithm with Mini-batch Sampling

Estelle Kuhn · Catherine Matias · Tabea Rebafka

Received: date / Accepted: date

Abstract To deal with very large datasets a mini-batch version of the Monte Carlo Markov Chain Stochastic Approximation Expectation-Maximization algorithm for general latent variable models is proposed. For exponential models the algorithm is shown to be convergent under classical conditions as the number of iterations increases. Numerical experiments illustrate the performance of the mini-batch algorithm in various models. In particular, we highlight that mini-batch sampling results in an important speed-up of the convergence of the sequence of estimators generated by the algorithm. Moreover, insights on the effect of the mini-batch size on the limit distribution are presented. Finally, we illustrate how to use mini-batch sampling in practice to improve results when a constraint on the computing time is given.

Keywords EM algorithm, mini-batch sampling, stochastic approximation, Monte Carlo Markov chain.

Mathematics Subject Classification (2010) 65C60 · 62F12

Estelle Kuhn
MaIAGE, INRA, Université Paris-Saclay
Jouy-en-Josas, France
E-mail: estelle.kuhn@inra.fr

Catherine Matias
Sorbonne Université, Université de Paris, CNRS
Laboratoire de Probabilités,
Statistique et Modélisation (LPSM)
Paris, France
E-mail: catherine.matias@math.cnrs.fr

Tabea Rebafka
Sorbonne Université, Université de Paris, CNRS
Laboratoire de Probabilités,
Statistique et Modélisation (LPSM)
Paris, France
E-mail: tabea.rebafka@upmc.fr

1 Introduction

On very large datasets the computing time of the classical expectation-maximization (EM) algorithm (Dempster et al., 1977) as well as its variants such as Monte Carlo EM, Stochastic Approximation EM, Monte Carlo Markov Chain-SAEM and others can be very long, since all data points are visited in every iteration. To circumvent this problem, a bunch of EM-type algorithms have been proposed, namely various mini-batch (Neal and Hinton, 1999; Liang and Klein, 2009; Karimi et al., 2018; Nguyen et al., 2019) and online (Titterington, 1984; Lange, 1995; Cappé and Moulines, 2009; Cappé, 2011) versions of the EM algorithm. They all consist in using only a part of the observations during one iteration in order to shorten computing time and accelerate convergence. While online algorithms process a single observation per iteration handled in the order of arrival, mini-batch algorithms use larger, randomly chosen subsets of observations. The size of these subsets of data is generally called mini-batch size. Choosing large mini-batch sizes entails long computing times, while very small mini-batch sizes as well as online algorithms may result in a loss of accuracy of the algorithm. This raises the question about the optimal mini-batch size that would achieve a compromise between accuracy and computing time. However this issue is generally overlooked.

In this article, we propose a mini-batch version of the MCMC-SAEM algorithm (Delyon et al., 1999; Kuhn and Lavielle, 2004). The original MCMC-SAEM algorithm is a powerful alternative to EM when the E-step is intractable. This is particularly interesting for nonlinear models or non-Gaussian models, where the unobserved data cannot be simulated exactly from the conditional distribution. Moreover, the MCMC-SAEM

algorithm is also more computing efficient than the MCMC-EM algorithm, since only a single instance of the latent variable is sampled at every iteration of the algorithm. Nevertheless, when the dimension of the latent variable is huge, the simulation step as well as the update of the sufficient statistic can be time consuming. From this point of view the here proposed mini-batch version is computationally more efficient than the original MCMC-SAEM, since at each iteration only a small proportion of the latent variables is simulated and only the corresponding data are visited to update the parameter estimates. For exponential models, we prove almost-sure convergence of the sequence of estimates generated by the mini-batch MCMC-SAEM algorithm under classical conditions as the number of iterations of the algorithm increases. We also conjecture an asymptotic normality result and the relation between the limiting covariance and the mini-batch size. Moreover, for various models we assess via numerical experiments the influence of the mini-batch size on the speed-up of the convergence at the beginning of the algorithm as well as its impact on the limit distribution of the estimates. Furthermore, we study the computing time of the algorithm and address the question of how to use mini-batch sampling in practice to improve results.

2 Latent variable model and algorithm

This section introduces the general latent variable model considered throughout this paper and the original MCMC-SAEM algorithm. Then the new mini-batch version of the MCMC-SAEM algorithm is presented.

2.1 Model and assumptions

Consider the common latent variable model with incomplete (observed) data $\mathbf{y}$ and latent (unobserved) variable $\mathbf{z}$. Denote n the dimension of the latent variable $\mathbf{z} = (z_1 \dots, z_n) \in \mathbb{R}^n$. In many models, n also corresponds to the number of observations, but it is not necessary that $\mathbf{z}$ and $\mathbf{y}$ have the same size or that each observation y_i depends only on a single latent component z_i, as it is for instance the case in the stochastic block model, Section 4.3.

Denote $\theta \in \Theta \subset \mathbb{R}^d$ the model parameter of the joint distribution of the complete data $(\mathbf{y}, \mathbf{z})$. In what follows, omitting all dependencies in the observations $\mathbf{y}$, which are considered as fixed realizations in the analysis, we assume that the complete-data likelihood function has the following form

$$f(\mathbf{z};\theta) = \exp\left\{-\psi(\theta) + \langle S(\mathbf{z}), \phi(\theta)\rangle\right\} c(\mathbf{z}),$$

where $\langle \cdot, \cdot \rangle$ is the scalar product, $S(\mathbf{z})$ denotes a vector of sufficient statistics of the model taking its values in some set $\mathcal{S}$ and ψ and ϕ are functions on Θ. The posterior distribution of the latent variables $\mathbf{z}$ given the observations is denoted by $\pi(\cdot;\theta)$.

2.2 Description of MCMC-SAEM algorithm

The original MCMC-SAEM algorithm proposed by Kuhn and Lavielle (2004) is appropriate for models, where the classical EM-algorithm cannot be applied due to difficulties in the E-step. In particular, in those models the conditional expectation $\mathrm{E}_{\theta_{k-1}}[S(\mathbf{z})]$ of the sufficient statistic under the current parameter value θ_{k-1} has no closed-form expression. In the MCMC-SAEM algorithm the quantity $\mathrm{E}_{\theta_{k-1}}[S(\mathbf{z})]$ is thus estimated by a stochastic approximation algorithm. This means that the classical E-step is replaced with a simulation step using a MCMC procedure combined with a stochastic approximation step. Here, we focus on a version where the MCMC part is a Metropolis-Hastings-within-Gibbs algorithm (Robert and Casella, 2004). More precisely, the k-th iteration of the classical MCMC-SAEM algorithm consists of the following three steps.

2.2.1 Simulation step

A new realization $\mathbf{z}_k$ of the latent variable is sampled from an ergodic Markov transition kernel $\Pi(\mathbf{z}_{k-1}, \cdot|\theta_{k-1})$, whose stationary distribution is the posterior distribution $\pi(\cdot;\theta_{k-1})$. In practice, this simulation is done by performing one iteration of a Metropolis-Hastings-within-Gibbs algorithm. That is, we consider a collection $(\Pi_i)_{1\le i\le n}$ of symmetric random walk Metropolis kernels defined on $\mathbb{R}^n$, where subscript i indicates that Π_i acts only on the i-th coordinate, see Fort et al. (2003). These kernels are applied successively to update the components of $\mathbf{z}$ one after the other. More precisely, let $(\mathbf{e}_i)_{1\le i\le n}$ be the canonical basis of $\mathbb{R}^n$. Then, for each $i \in \{1, \dots, n\}$ starting from the n-vector $\mathbf{z} = (z_1, \dots, z_n)$, the proposal in the direction of $\mathbf{e}_i$ is given by $\mathbf{z} + x\mathbf{e}_i$, where $x \in \mathbb{R}$ is sampled from a symmetric increment density q_i. This proposal is then accepted with probability $\min\{1, \pi(\mathbf{z} + x\mathbf{e}_i;\theta_{k-1})/\pi(\mathbf{z};\theta_{k-1})\}$.

2.2.2 Stochastic approximation step

The approximation of the sufficient statistic is updated by

$$\mathbf{s}_k = (1-\gamma_k)\mathbf{s}_{k-1} + \gamma_k S(\mathbf{z}_k), \tag{1}$$

where $(\gamma_k)_{k\ge1}$ is a decreasing sequence of positive stepsizes such that $\sum_{k=1}^{\infty}\gamma_k = \infty$ and $\sum_{k=1}^{\infty}\gamma_k^2 < \infty$. That

is, the current approximation $\mathbf{s}_k$ of the sufficient statistic is a weighted mean of its previous value $\mathbf{s}_{k-1}$ and the value of the sufficient statistic $S(\mathbf{z}_k)$ evaluated on the current value of the simulated latent variable $\mathbf{z}_k$.

2.2.3 Maximization step

The model parameter θ is updated by $\theta_k = \hat{\theta}(\mathbf{s}_k)$

with $\hat{\theta}(\mathbf{s}) = \arg\max_{\theta \in \Theta} L(\theta; \mathbf{s})$,

where $L(\theta; \mathbf{s}) = -\psi(\theta) + \langle \mathbf{s}, \phi(\theta) \rangle$.

Depending on the model the maximization problem may have a closed-form solution or not.

2.3 Mini-batch MCMC-SAEM algorithm

When the dimension n of the latent variable $\mathbf{z}$ is large, the simulation step can be very time-consuming. Indeed, simulating *all* latent components z_i at every iteration is costly in time. Thus, according to the spirit of other mini-batch algorithms, updating only a part of the latent components may speed up the computing time and also the convergence of the algorithm. With this idea in mind, denote $0 < \alpha \leq 1$ the average proportion of components of the latent variable $\mathbf{z}$ that are updated during one iteration.

Furthermore, depending on the model, the evaluation of the sufficient statistic $S(\mathbf{z}_k)$ on the current latent variable $\mathbf{z}_k$ can be accelerated as only a part of the components of $\mathbf{z}_k$ have changed. This brings a further gain in computing time.

2.3.1 Mini-batch simulation step

In the mini-batch version of the MCMC-SAEM algorithm the simulation step consists of two parts. First, select the indices of the components z_i of the latent variable that will be updated. That is, we sample the number r_k of indices from a binomial distribution $\text{Bin}(n, \alpha)$ and then randomly select r_k indices among $\{1, \ldots, n\}$ without replacement. Denote $\mathcal{I}_k$ this set of selected indices at iteration k. Second, instead of sampling all components $z_{k,i}$, we only sample from the Metropolis kernels Π_i for $i \in \mathcal{I}_k$ to update only the components $z_{k,i}$ with index $i \in \mathcal{I}_k$.

2.3.2 Stochastic approximation step

Again this step consists in updating the sufficient statistic $\mathbf{s}_k$ according to Equation (1). However, the naive evaluation of the sufficient statistic $S(\mathbf{z}_k)$ generally involves all data $\mathbf{y}$ and thus is time-consuming on large datasets. Though, in most models it is computationally much more efficient to derive the value of $S(\mathbf{z}_k)$ from its previous value $S(\mathbf{z}_{k-1})$ by correcting only for the terms that involve recently updated latent components. In general, this amounts to using only a small part of the data and thus speeds up computing. An example is detailed in Section 4.3.

Algorithm 1 Mini-batch MCMC-SAEM

Input: data $\mathbf{y}$.
Initialization: Choose initial values θ_0, $\mathbf{s}_0$, $\mathbf{z}_0$.
Set $k = 1$.
while not converged **do**
  Sample $r_k \sim \text{Bin}(n, \alpha)$.
  Sample r_k indices from $\{1, \ldots, n\}$, denoted by $\mathcal{I}_k$.
  Set $\mathbf{z}_k = \mathbf{z}_{k-1}$
  for $i \in \mathcal{I}_k$ **do**
    Sample $\mathbf{z} \sim \Pi_i(\mathbf{z}_k, \cdot|\theta_{k-1})$.
    Set $\mathbf{z}_k = \mathbf{z}$.
  end for
  $\mathbf{s}_k = (1 - \gamma_k)\mathbf{s}_{k-1} + \gamma_k S(\mathbf{z}_k)$.
  Update parameter $\theta_k = \hat{\theta}(\mathbf{s}_k)$.
  Increment k.
end while

2.3.3 Maximization step

This step is identical to the one in the original algorithm. It does not depend on the mini-batch proportion α in any way: the formulae for the update of the parameter estimates are identical and the computing time of the M-step is the same for any α.

2.3.4 Initialization

Initial values θ_0, $\mathbf{s}_0$ and $\mathbf{z}_0$ for the model parameter, the sufficient statistic and the latent variable, respectively, have to be chosen by the user or at random.

See Algorithm 1 for a complete description of the algorithm.

3 Convergence of the algorithm

In this section we show that, under appropriate assumptions, the mini-batch MCMC-SAEM algorithm converges as the number of iterations increases. Note that we consider convergence of the algorithm for a fixed dataset $\mathbf{y}$ when the number of iterations tends to infinity, and not statistical convergence where the sample size grows. Basically, the assumptions are classical conditions that ensure the convergence of the original MCMC-SAEM algorithm. So roughly, our theorem says that if the batch MCMC-SAEM algorithm converges, so does the mini-batch version for any mini-batch proportion α.

Compared to the classical MCMC-SAEM algorithm, the mini-batch version involves an additional stochastic part that comes from the selection of indices of the latent components that are to be updated. This additional randomness is governed by the value of the mini-batch proportion α.

We also present arguments to explain the impact of the mini-batch proportion α onto the limit distribution of the sequence generated by the algorithm.

3.1 Equivalent descriptions

The above description of the simulation step is convenient for achieving maximal computing efficiency. We now focus on a mathematically equivalent framework that underlines the fact that the mini-batch MCMC-SAEM algorithm formally belongs to the family of classical MCMC-SAEM algorithms.

For two kernels P_1 and P_2, we denote their composition by

$$P_2 \circ P_1(\mathbf{z}, \mathbf{z}') = \int_{\tilde{\mathbf{z}}} P_2(\tilde{\mathbf{z}}, \mathbf{z}') P_1(\mathbf{z}, d\tilde{\mathbf{z}}).$$

With this notation at hand, the Metropolis-Hastings-within-Gibbs uses the kernel $\Pi = \Pi_n \circ \cdots \circ \Pi_1$. Now, to describe the mini-batch simulation step in terms of a Markov kernel, we first introduce kernel $\Pi_{\alpha,i}$, which is a mixture of the original kernel Π_i and the identity kernel Id, defined as

$$\Pi_{\alpha,i}(\mathbf{z}, \mathbf{z}'|\theta) = \\ \alpha \Pi_i(\mathbf{z}, (z_1, \ldots, z_i', \ldots, z_n)|\theta) + (1-\alpha)\mathrm{Id}(\mathbf{z}, \mathbf{z}').$$

Hence, the mini-batch simulation step corresponds to generating a latent vector $\mathbf{z}$ according to the Markov kernel $\Pi_\alpha = \Pi_{\alpha,n} \circ \cdots \circ \Pi_{\alpha,1}$. Indeed, Π_α can also be written as

$$\Pi_\alpha(\mathbf{z}, \mathbf{z}'|\theta) = \\ \sum_{k=0}^{n} \alpha^k (1-\alpha)^{n-k} \sum_{1 \le i_1 < \cdots < i_k \le n} (\Pi_{i_k} \circ \cdots \circ \Pi_{i_1})(\mathbf{z}, \mathbf{z}'|\theta).$$

That means that Π_α is a mixture of compositions of the original kernels Π_i and the identity kernel. In other words, the mini-batch MCMC-SAEM algorithm corresponds to the family of classical MCMC-SAEM algorithms with a particular choice of the transition kernel. Note that Π_α can also be interpreted as a mixture over different trajectories (the choice of indices $i_1 < \cdots < i_k$ to be updated) of Metropolis-Hastings-within-Gibbs kernels acting on a part of the latent vector $\mathbf{z}$.

We now give a third mathematically equivalent description of the simulation step, which will be appropriate for the analysis of the theoretical properties of the algorithm. In the k-th mini-batch simulation step, we sample for every $i \in \{1, \ldots, n\}$ a Bernoulli random variable $U_{k,i}$ with parameter α. So $U_{k,i}$ indicates whether the latent variable $\mathbf{z}_{k-1,i}$ is updated at iteration k or not. Next, we sample a realization $\tilde{\mathbf{z}}_{k,i}$ from the transition kernel Π_i and set

$$\mathbf{z}_{k,i} = U_{k,i}\tilde{\mathbf{z}}_{k,i} + (1 - U_{k,i})\mathbf{z}_{k-1,i}. \tag{2}$$

In particular, we see from this formula that, for $\alpha = 1$, the sequence $(\mathbf{z}_k)_{k\ge1}$ generated by the batch algorithm is a Markov chain with transition kernel $\Pi = \Pi_n \circ \cdots \circ \Pi_1$, what has already been mentioned above.

Fort et al. (2003) establish results on the geometric ergodicity of hybrid samplers and in particular for the random-scan Gibbs sampler. The latter is defined as $n^{-1}\sum_{i=1}^n \Pi_i$, where each Π_i is a kernel on $\mathbb{R}^n$ acting only on the i-th component. More generally, the random-scan Gibbs sampler may be defined as $\sum_{i=1}^n a_i \Pi_i$, where $(a_1, \ldots, a_n)$ is a probability distribution. This means that at each step of the algorithm, only one component i is drawn from the probability distribution $(a_1, \ldots, a_n)$ and then updated. These probabilities may be chosen uniformly ($a_i = 1/n$) or, for example, can be used to favor a component that is more difficult to explore. We generalize the results of Fort et al. (2003) to a setup, where at each step k kernel $\bar{\Pi}_\alpha$ is used that is iterated from a random-scan Gibbs sampler $\widetilde{\Pi}_\alpha$ as follows

$$\bar{\Pi}_\alpha(\cdot, \cdot|\theta_k) = \widetilde{\Pi}_\alpha(\cdot, \cdot|\theta_k)^{\sum_i U_{k,i}}, \tag{3}$$

with

$$\widetilde{\Pi}_\alpha(\cdot, \cdot|\theta_k) = \\ \begin{cases} \left(\sum_{i=1}^n U_{k,i}\right)^{-1} \sum_{i=1}^n U_{k,i} \Pi_i(\cdot, \cdot|\theta_k) & \text{if } \sum_{i=1}^n U_{k,i} \ge 1, \\ \mathrm{Id} & \text{else} \end{cases}$$

Note that this is not exactly the kernel corresponding to the algorithm described above, as the same component i can possibly be updated more than once during the same iteration. Nonetheless, we neglect this effect and establish our result for the algorithm based on kernel $\bar{\Pi}_\alpha$.

3.2 Assumptions and convergence result

Assume that the random variables $\mathbf{s}_0, \mathbf{z}_1, \mathbf{z}_2, \ldots$ are defined on the same probability space $(\Omega, \mathcal{A}, P)$. We denote $\mathcal{F} = \{\mathcal{F}_k\}_{k\ge0}$ the increasing family of σ-algebras generated by the random variables $\mathbf{s}_0, \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k$. Consider the following regularity conditions.

(**M1**) The parameter space Θ is an open subset of $\mathbb{R}^d$. The complete-data likelihood function is given by

$$f(\mathbf{z};\theta) = \exp\left\{-\psi(\theta) + \langle S(\mathbf{z}), \phi(\theta)\rangle\right\} c(\mathbf{z}),$$

where S is a continuous function on $\mathbb{R}^n$ taking its values in an open subset $\mathcal{S}$ of $\mathbb{R}^m$. Moreover, the convex hull of $S(\mathbb{R}^n)$ is included in $\mathcal{S}$, and for all $\theta \in \Theta$,

$$\int |S(\mathbf{z})| \pi(\mathbf{z};\theta) d\mathbf{z} < \infty.$$

(**M2**) The functions ψ and ϕ are twice continuously differentiable on Θ.

(**M3**) The function $\bar{s} : \Theta \to \mathcal{S}$ defined as

$$\bar{\mathbf{s}}(\theta) = \int S(\mathbf{z}) \pi(\mathbf{z};\theta) d\mathbf{z},$$

is continuously differentiable on Θ.

(**M4**) The observed-data log-likelihood function $\ell : \Theta \to \mathbb{R}$ defined as

$$\ell(\theta) = \log \int f(\mathbf{z};\theta) d\mathbf{z},$$

is continuously differentiable on Θ and

$$\partial_\theta \int f(\mathbf{z};\theta) d\mathbf{z} = \int \partial_\theta f(\mathbf{z};\theta) d\mathbf{z}.$$

(**M5**) Define $L : \mathcal{S} \times \Theta \to \mathbb{R}$ as $L(\mathbf{s};\theta) = -\psi(\theta) + \langle \mathbf{s}, \phi(\theta)\rangle$. There exists a continuously differentiable function $\widehat{\theta} : \mathcal{S} \to \Theta$, such that, for any $\mathbf{s} \in \mathcal{S}$ and any $\theta \in \Theta$,

$$L(\mathbf{s}; \widehat{\theta}(\mathbf{s})) \geq L(\mathbf{s};\theta).$$

We now introduce the usual conditions that ensure convergence of the SAEM procedure.

(**SAEM1**) For all $k \in \mathbb{N}$, $\gamma_k \in [0,1]$, $\sum_{k=1}^{\infty} \gamma_k = \infty$ and $\sum_{k=1}^{\infty} \gamma_k^2 < \infty$.

(**SAEM2**) The functions $\ell : \Theta \to \mathbb{R}$ and $\widehat{\theta} : \mathcal{S} \to \Theta$ are m times differentiable, where we recall that $\mathcal{S}$ is an open subset of $\mathbb{R}^m$.

For any $\mathbf{s} \in \mathcal{S}$, we define $H_{\mathbf{s}}(\mathbf{z}) = S(\mathbf{z}) - \mathbf{s}$ and its expectation with respect to the posterior distribution $\pi(\cdot; \hat{\theta}(\mathbf{s}))$ denoted by $h(\mathbf{s}) = \mathrm{E}_{\hat{\theta}(\mathbf{s})}[S(\mathbf{z})] - \mathbf{s}$. For any $\rho > 0$, denote $V_\rho(\mathbf{z}) = \sup_{\theta \in \Theta} [\pi(\mathbf{z};\theta)]^\rho$. We consider the following additional assumptions as done in Fort et al. (2003).

(**H1**) There exists a constant M_0 such that

$$\mathcal{L} = \left\{\mathbf{s} \in \mathcal{S}, \langle \nabla \ell(\hat{\theta}(\mathbf{s})), h(\mathbf{s})\rangle = 0\right\} \subset \{\mathbf{s} \in \mathcal{S}, -\ell(\hat{\theta}(\mathbf{s})) < M_0\}.$$

In addition, there exist $M_1 \in (M_0, \infty]$ such that $\{\mathbf{s} \in \mathcal{S}, -\ell(\hat{\theta}(\mathbf{s})) \leq M_1\}$ is a compact set.

(**H2**) The family $\{q_i\}_{1 \leq i \leq n}$ of symmetric densities is such that, for $i = 1, \ldots, n$, there exist constants $\eta_i > 0$ and $\delta_i < \infty$ such that $q_i(x) > \eta_i$ for all $|x| < \delta_i$.

(**H3**) There are constants δ and Δ with $0 \leq \delta \leq \Delta \leq \infty$ such that

$$\inf_{i=1,\ldots,n} \int_\delta^\Delta q_i(x) dx > 0,$$

and for any sequence $\{\mathbf{z}^j\}$ with $\lim_{j\to\infty} |\mathbf{z}^j| = \infty$, a subsequence $\{\check{\mathbf{z}}^j\}$ can be extracted with the property that, for some $i \in \{1, \ldots, n\}$, for any $x \in [\delta, \Delta]$ and any $\theta \in \Theta$,

$$\lim_{j\to\infty} \frac{\pi(\check{\mathbf{z}}^j;\theta)}{\pi(\check{\mathbf{z}}^j - \mathrm{sign}(z_i^j) x \mathbf{e}_i;\theta)} = 0 \quad \text{and}$$
$$\lim_{j\to\infty} \frac{\pi(\check{\mathbf{z}}^j + \mathrm{sign}(z_i^j) x \mathbf{e}_i;\theta)}{\pi(\check{\mathbf{z}}^j;\theta)} = 0.$$

(**H4**) There exist $C > 1$, $\rho \in (0,1)$ and $\theta_0 \in \Theta$ such that, for all $\mathbf{z} \in \mathbb{R}^n$,

$$|S(\mathbf{z})| \leq C \pi(\mathbf{z};\theta_0)^{-\rho}.$$

To state our convergence result, we consider the version of the algorithm with truncation on random boundaries studied by Andrieu et al. (2005). This additional projection step ensures in particular the stability of the algorithm for the theoretical analysis and is only a technical tool for the proof without any practical consequences.

Theorem 1 *Assume that the conditions* (**M1**)–(**M5**), (**SAEM1**), (**SAEM2**) *and* (**H1**)–(**H4**) *hold. Let* $0 < \alpha \leq 1$ *and* $(\theta_k)_{k\geq 1}$ *be a sequence generated by the mini-batch MCMC-SAEM algorithm with corresponding Markov kernel* $\bar{\Pi}_\alpha(\cdot,\cdot|\theta)$*. Then*

$$\lim_{k\to\infty} d\left(\theta_k, \{\theta, \nabla \ell(\theta) = 0\}\right) = 0,$$

where $d(x, A) = \inf\{y \in A, |x - y|\}$*, that is,* $(\theta_k)_{k\geq 1}$ *converges to the set of critical points of the observed likelihood* $\ell(\theta)$ *as the number of iterations increases.*

Proof The proof consists of two steps. First, we prove the convergence of the sequence of sufficient statistics $(\mathbf{s}_k)_{k\geq 1}$ towards the set of zeros of function h using Theorem 5.5 in Andrieu et al. (2005). Second, following the usual reasoning for EM-type algorithms, described for instance in Delyon et al. (1999), we deduce that the sequence $(\theta_k)_{k\geq 1}$ converges to the set of critical points of the observed data log-likelihood ℓ.

First step. In order to apply Theorem 5.5 in Andrieu et al. (2005), we need to establish that their conditions (A1) to (A4) are satisfied. In what follows, (A1) to (A4) refer to the conditions stated in Andrieu et al. (2005). First, note that under our assumptions (**H1**), (**M1**)–(**M5**) and (**SAEM2**), condition (A1) is satisfied. Indeed, this is a consequence of Lemma 2 in Delyon et al. (1999). To establish (A2) and (A3), as suggested in Andrieu et al. (2005), we establish their drift conditions (DRI), see Proposition 6.1 in Andrieu et al. (2005). We first focus on establishing (DRI1) in Andrieu et al. (2005). To this aim, we rely on Fort et al. (2003) that establish results for the random-scan Metropolis sampler. In their context, they consider a sampler $\Pi = n^{-1}\sum_{i=1}^{n}\Pi_i$. We generalize their results to our setup according to (3). Following the lines of the proof of Theorem 2 in Fort et al. (2003), we can show that Equations (6.1) and (6.3) appearing in the drift condition (DRI1) in Andrieu et al. (2005) are satisfied when (**H2**)–(**H3**) hold. Indeed following the strategy developed in Allassonnière et al. (2010), we first establish Equations (6.1) and (6.3) using a drift function depending on θ, namely $V_\theta(\mathbf{z}) = \pi(\mathbf{z};\theta)^{-\rho}$, where ρ is given by (**H4**). Then we define the common drift function V as follows. Let $\theta_0 \in \Theta$ and ρ be given in (**H4**) and define $V(\mathbf{z}) = \pi(\mathbf{z};\theta_0)^{-\rho}$. Then for any compact $\mathcal{K} \in \Theta$, there exist two positive constants $c_\mathcal{K}$ and $C_\mathcal{K}$ such that for all $\theta \in \mathcal{K}$ and for all $\mathbf{z}$, we get $c_\mathcal{K} V(\mathbf{z}) \leq \pi(\mathbf{z};\theta)^{-\rho} \leq C_\mathcal{K} V(\mathbf{z})$. We then establish Equations (6.1) and (6.3) for this drift function V. Moreover, using Proposition 1 and Proposition 2 in Fort et al. (2003) we obtain that Equation (6.2) in (DRI1) from Andrieu et al. (2005) holds. Under assumption (**H4**) we have the first part of (DRI2) in Andrieu et al. (2005). The second part is true in our case with $\beta = 1$. Finally, (DRI3) in Andrieu et al. (2005) holds in our context with $\beta = 1$, since $\mathbf{s} \mapsto \hat{\theta}(\mathbf{s})$ is twice continuously differentiable and thus Lipschitz on any compact set. To prove this, we decompose the space in an acceptance region and a rejection region and consider the integral over four sets leading to different expressions of the acceptance ratio (see, for example, the proof of Lemma 4.7 in Fort et al., 2015). This implies that (DRI) and therefore (A2)–(A3) in Andrieu et al. (2005) are satisfied. Notice that (**SAEM1**) ensures (A4). This concludes the first step of the proof.

Second step. As the function $\mathbf{s} \mapsto \hat{\theta}(\mathbf{s})$ is continuous, the second step is immediate by applying Lemma 2 in Delyon et al. (1999).

3.3 Limit distribution

The theoretical study of the impact of the mini-batch proportion α on the limit distribution of the sequence $(\theta_k)_{k\geq 1}$ generated by the mini-batch MCMC-SAEM algorithm is involved, and here we only present some heuristic arguments. We conjecture that, under reasonable assumptions, $(\theta_k)_{k\geq 1}$ is asymptotically normal at rate $1/\sqrt{k}$ and the limiting covariance matrix, say V_α, depends on the mini-batch proportion α in the following form

$$V_\alpha = \frac{2-\alpha}{\alpha} V_1,$$

where V_1 denotes the limiting covariance of the batch algorithm. This formula is coherent with the expected behavior with respect to the mini-batch proportion. Namely, the limit variance is monotone in α, for $\alpha = 1$ we recover the limit variance of the batch algorithm, and, when α vanishes, the limit variance tends to infinity. Numerical experiments in Sections 4.3 and 4.4 support this conjecture.

The general approach to establish asymptotic normality of $(\theta_k)_{k\geq 1}$ consists in establishing asymptotic normality of the sequence of sufficient statistics $(\mathbf{s}_k)_{k\geq 1}$ and then applying the delta method. Now consider the simple case where the model has a single latent component, that is, $n = 1$. We rewrite Equation (1) as

$$\mathbf{s}_k = \mathbf{s}_{k-1} + \gamma_k h(\mathbf{s}_k) + \gamma_k \eta_k,$$

where $\eta_k = S(\mathbf{z}_k) - \mathrm{E}_{\theta_{k-1}}[S(\mathbf{z})]$, and note that $S(\mathbf{z}_k) = U_k S(\tilde{\mathbf{z}}_k) + (1-U_k) S(\mathbf{z}_{k-1})$. In general, the principal contribution to the limit variance comes from the term

$$\frac{1}{\sqrt{k}}\sum_{l=1}^{k}\eta_l = \frac{1}{\sqrt{k}}\sum_{l=1}^{k} N_{l,k}\left(S(\tilde{\mathbf{z}}_l) - \mathrm{E}_{\theta_{l-1}}[S(\mathbf{z})]\right) + \frac{1}{\sqrt{k}}\sum_{l=1}^{k}(N_{l,k}-1)\mathrm{E}_{\theta_{l-1}}[S(\mathbf{z})], \tag{4}$$

where

$$N_{l,k} = U_l \sum_{j=l}^{k}\left\{\prod_{m=1}^{j}(1-U_{l+m})\right\},$$

The quantity $N_{l,k}$ equals zero if there is no update of the unique latent component at iteration l. Otherwise, $N_{l,k}$ is the number of iterations until the next update after the one at iteration l. The random variable $N_{l,k}$ takes its values in $\{0,1,\ldots,k-l+1\}$ and it can be shown that

$$\mathbb{P}(N_{l,k}=m) = \begin{cases} 1-\alpha, & m=0 \\ \alpha^2(1-\alpha)^{m-1}, & m=1,\ldots,k-l \\ \alpha(1-\alpha)^{k-l}, & m=k-l+1 \end{cases}.$$

Another important property is that $\sum_{l=1}^{k} N_{l,k} = k$ *a.s.*

It can be shown that the second term in the right-hand side of (4) tends to zero in probability when k goes to infinity. To analyze the first term, we consider the simple setting where for all $k \geq 1$, $\theta_k = \theta^*$, where θ^* is constant, as e.g. a critical point of the observed likelihood, and $\tilde{\mathbf{z}}_k$ are simulated from the conditional distribution $\pi(\cdot;\theta^*)$. In this case, conditionally to the Bernoulli indicators of updates $(U_k)_{k\geq 1}$, the central limit theorem can be applied to the first term. For the conditional variance of the first term in Equation (4) we obtain

$$\begin{aligned}&\mathrm{Var}\left(\frac{1}{\sqrt{k}}\sum_{l=1}^{k} N_{l,k}\left(S(\tilde{\mathbf{z}}_l) - \mathrm{E}_{\theta_{l-1}}[S(\mathbf{z})]\right)\middle|(U_k)_{k\geq 1}\right)\\ &= \frac{1}{\sqrt{k}}\sum_{l=1}^{k} N_{l,k}^2 V_1,\end{aligned}$$

where V_1 is the variance of $S(\mathbf{z})$ with $\mathbf{z} \sim \pi(\cdot;\theta^*)$. Further computations yield that the expectation taken over $(U_k)_{k\geq 1}$ gives

$$\lim_{k\to\infty} \mathrm{E}\left[\frac{1}{\sqrt{k}}\sum_{l=1}^{k} N_{l,k}^2\right] = \frac{2-\alpha}{\alpha}.$$

The main difficulty for generalizing this approach to $n > 1$ arises from two facts. First, the components of $\mathbf{z}_k$ are not updated simultaneously, but at different iterations. Second, the sufficient statistic may not be linear in $\mathbf{z}$. More precisely, when $\mathbf{z}$ is a vector, $N_{l,k}$ is also a vector and an equivalent of (4) is not immediate.

4 Numerical experiments

We carry out various numerical experiments in a nonlinear mixed effects model, a Bayesian deformable template model, the stochastic block model and a frailty model, to illustrate the performance and the properties of the proposed mini-batch MCMC-SAEM algorithm and the potential gain in efficiency and accuracy.

4.1 Nonlinear mixed model for pharmacokinetic study

Clinical pharmacokinetic studies aim at analyzing the evolution of the concentration of a given drug in the blood of an individual over a given time interval after absorbing the drug. In this section a classical one-compartment model is considered.

4.1.1 Model

The model presented in Davidian and Giltinan (1995) serves to analyze the kinetic of the drug theophylline used in therapy for respiratory diseases. For $i = 1, \ldots, n$ and $j = 1, \ldots, J$, we define

$$y_{ij} = h(V_i, \mathrm{ka}_i, \mathrm{Cl}_i) + \varepsilon_{ij}$$

with

$$h(V_i, \mathrm{ka}_i, \mathrm{Cl}_i) = \frac{d_i \mathrm{ka}_i}{V_i \mathrm{ka}_i - \mathrm{Cl}_i}\left[\mathrm{e}^{-\mathrm{Cl}_i t_{ij}/V_i} - \mathrm{e}^{-\mathrm{ka}_i t_{ij}}\right]$$

where the observation y_{ij} is the measure of drug concentration on individual i at time t_{ij}. The drug dose administered to individual i is denoted d_i. The parameters for individual i are the volume V_i of the central compartment, the constant ka_i of the drug absorption rate, and the drug's clearance Cl_i. The random measurement error is denoted by ε_{ij} and supposed to have a centered normal distribution with variance σ^2. For the individual parameters V_i, ka_i and Cl_i log-normal distributions are considered given by

$$\begin{aligned}\log V_i &= \log(\mu_V) + z_{i,1},\\ \log \mathrm{ka}_i &= \log(\mu_{\mathrm{ka}}) + z_{i,2},\\ \log \mathrm{Cl}_i &= \log(\mu_{\mathrm{Cl}}) + z_{i,3},\end{aligned}$$

where $z_i = (z_{i,1}, z_{i,2}, z_{i,3})$ are independent latent random variables following a centered normal distribution with variance $\Omega = \mathrm{diag}(\omega_V^2, \omega_{\mathrm{ka}}^2, \omega_{\mathrm{Cl}}^2)$. Then the model parameters are $\theta = (\mu_V, \mu_{\mathrm{ka}}, \mu_{\mathrm{Cl}}, \omega_V^2, \omega_{\mathrm{ka}}^2, \omega_{\mathrm{Cl}}^2, \sigma^2)$.

4.1.2 Algorithm

We implement the minibatch MCMC-SAEM algorithm presented in Section 2.3. In the simulation step we use the following sampling procedure. Let $\mathcal{I}_k$ be the subset of indices of latent variable components z_i that have to be updated at iteration k. For each $i \in \mathcal{I}_k$, we use a Metropolis-Hastings procedure: first, for $l \in \{1, 2, 3\}$ draw a candidate $\tilde{z}_{k,i,l}$ from the normal distribution $\mathcal{N}(z_{k-1,i,l}, \eta_l)$, with $\eta_1 = 0.01$, $\eta_2 = 0.02$ and $\eta_3 = 0.03$, chosen to get good mean acceptance rates. Then compute the acceptance ratio $\rho_{k,i,l} = \rho(z_{k-1,i,l}, \tilde{z}_{k,i,l})$ of the usual Metropolis-Hastings procedure. Finally, draw a realization $\omega_{k,i,l}$ of the uniform distribution $U[0,1]$ and set $z_{k,i,l} = \tilde{z}_{k,i,l}$ if $\omega_{k,i,l} < \rho_{k,i,l}$, and $z_{k,i,l} = z_{k-1,i,l}$ otherwise.

In the next step, we compute the stochastic approximation of sufficient statistics of the model taking value in $\mathbb{R}^7$ according to

$$s_k = (1-\gamma_k)s_{k-1} + \gamma_k S(\mathbf{z}_k),$$

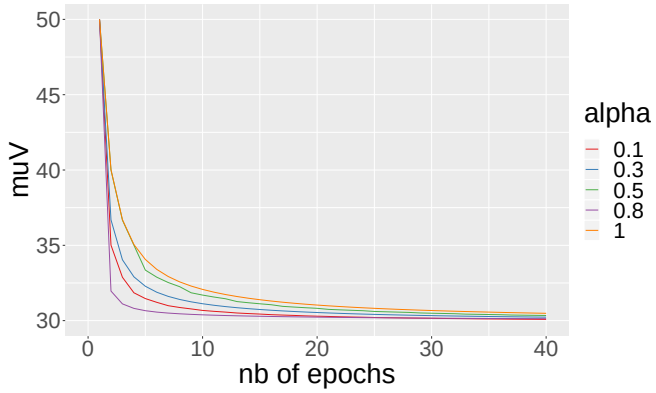

Fig. 1 Estimates of the parameter μ_V using mini-batch MCMC-SAEM with $\alpha \in \{0.1, 0.3, 0.5, 0.8, 1\}$ as a function of the number of epochs.

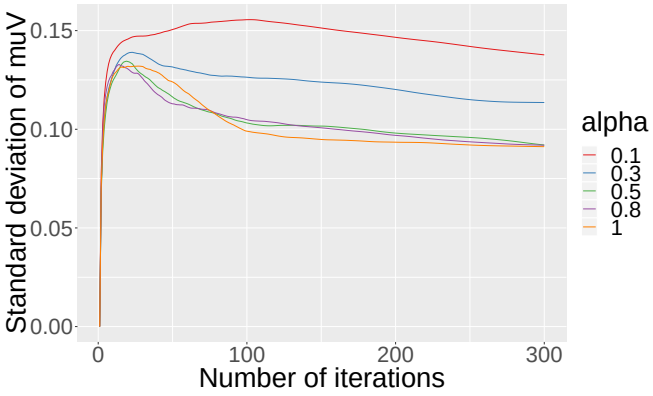

Fig. 2 Sample standard deviation of the estimate of parameter μ_V using mini-batch MCMC-SAEM with $\alpha \in \{0.1, 0.3, 0.5, 0.8, 1\}$ as a function of the number of iterations.

where

$$S(\mathbf{z}_k) = \left(\frac{1}{n}\sum_i z_{i,1}, \frac{1}{n}\sum_i z_{i,2}, \frac{1}{n}\sum_i z_{i,3}, \frac{1}{n}\sum_i z_{i,1}^2, \right.$$
$$\left. \frac{1}{n}\sum_i z_{i,2}^2, \frac{1}{n}\sum_i z_{i,3}^2, \frac{1}{nJ}\sum_{i,j} (y_{ij} - h(V_i, \mathrm{ka}_i, \mathrm{Cl}_i))^2 \right)$$

and where the sequence $(\gamma_k)_{k\geq 1}$ is chosen such that $0 < \gamma_k < 1$ for all k, $\sum \gamma_k = \infty$ and $\sum \gamma_k^2 < \infty$.

Finally the maximization step is performed using explicit solutions given by the following equations

$$\mu_{V,k} = \exp(s_{k,1}); \qquad \omega_{V,k}^2 = s_{k,4} - s_{k,1}^2;$$
$$\mu_{\mathrm{ka},k} = \exp(s_{k,2}); \qquad \omega_{\mathrm{ka},k}^2 = s_{k,5} - s_{k,2}^2;$$
$$\mu_{\mathrm{Cl},k} = \exp(s_{k,3}); \qquad \omega_{\mathrm{Cl},k}^2 = s_{k,6} - s_{k,3}^2;$$
$$\sigma_k^2 = s_{k,7}.$$

For more technical details on the implementation, we refer to (Kuhn and Lavielle, 2005).

4.1.3 Numerical results

In a simulation study we generate one dataset from the above model with the following parameter values $n = 1000, J = 10, \mu_V = 30, \mu_{\mathrm{ka}} = 1.8, \mu_{\mathrm{Cl}} = 3.5, \omega_V = 0.02, \omega_{\mathrm{ka}} = 0.04, \omega_{\mathrm{Cl}} = 0.06, \sigma^2 = 2$. For all individuals i the same dose $d_i = 320$ and the time points $t_{ij} = j$ are used.

Then we estimate the model parameters by both the original MCMC-SAEM algorithm and our mini-batch version. The step sizes are set to $\gamma_k = 1$ for $1 \leq k \leq 50$ and $\gamma_k = (k-50)^{-0.6}$ otherwise. Several mini-batch proportions, namely $\alpha \in \{0.1, 0.3, 0.5, 0.8, 1\}$, are considered. To be more precise, the estimation task is repeated 100 times on the same fixed dataset for all considered algorithms. The results for parameter μ_V are shown in Figures 1 and 2. The results for the other parameters are very similar and therefore omitted.

Figure 1 shows the evolution of the precision of the running mean of the estimates $\bar{\mu}_{V,k} = \sum_{l=1}^{k} \mu_{V,l}/k$ of parameter component μ_V as a function of the number of epochs for different values of the proportion α. An epoch is the average number of iterations required to update n latent components. That is, one epoch corresponds in average to $1/\alpha$ iterations of the mini-batch algorithm with proportion α. This means that parameter estimators are compared when the different algorithms have spent approximately the same time in the simulation step, and, due to the dependency structure of the nonlinear mixed model, when the algorithms have visited approximately the same amount of data. That is, an epoch takes approximately the same computing time for any proportion α. So Figure 1 compares estimates at comparable computing times. It is obvious that for all algorithms estimation improves when the number of epochs increases. Moreover, and more importantly, the rate of convergence depends on the mini-batch proportion: the smaller α, the faster the convergence to the target value $\mu_V = 30$. Here, the fastest convergence is obtained with the smallest mini-batch proportion, that is, $\alpha = 0.1$. For instance, to attain the precision obtained within 5 epochs with $\alpha = 0.1$, we need at least 25 epochs with the batch algorithm $\alpha = 1$.

This acceleration of convergence at the beginning of the algorithm induced by mini-batch sampling is characteristic for mini-batch sampling in any EM-type algorithm. Let us give an intuitive explanation of this characteristic phenomenon. In general, the initial values of the algorithm are far away from the unknown target values. So, during the first iteration of the batch algorithm, many time-consuming computations are done using a very bad value θ_0. Only at the very end of the

first iteration, the parameter estimate is updated to a slightly better value θ_1. During the same time, a mini-batch algorithm with small α performs some computations with the same bad value θ_0, but after a short time already, the M-step is attained for the first time. As only a couple of latent components have been updated and only a few data points have been visited, the new value θ_1 may be only a very slight correction of θ_0, but, nevertheless, it is a move into the right direction and the next iteration is performed using a slightly better value than in the previous one. Hence, performing mini-batch sampling consists in making many small updates of the θ, while in the same time the batch algorithm only makes very few updates. Metaphorically speaking, the batch algorithm makes long and time-consuming steps, but these steps are not necessarily directed into the best direction, whereas the mini-batch version makes plenty small and quick steps, correcting its direction after every step. As a whole, the mini-batch strategy leads to much better results as illustrated in Figure 1.

Figure 2 presents for different values of the proportion α the estimates of the empirical standard deviation with respect to the number of iterations. We observe that as the number of iterations increases, the standard deviations are lower than for higher values of α. This illustrates in particular that including more data in the inference task leads to more accurate estimation results. This is indeed very intuitive. Therefore an optimal choice of α should achieve a trade-off between speeding up the convergence and involving enough data in the process to get accurate estimates.

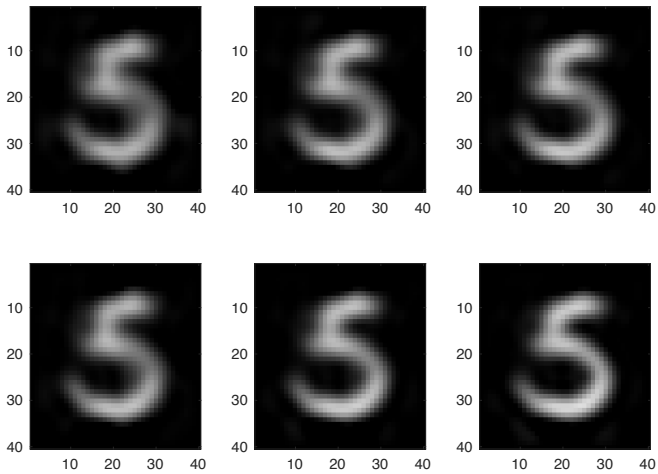

Fig. 3 Estimation of the template: first row : using batch MCMC-SAEM ; second row : using mini-batch MCMC-SAEM with $\alpha = 0.1$; columns correspond to 1, 2 and 3 epochs, respectively.

4.2 Deformable template model for image analysis

In this section an example on handwritten digits illustrates the benefits of using mini-batch sampling. We consider the dense deformation template model for image analysis that was first introduced in Allassonnière et al. (2007). This model considers observed images as deformations of a common reference image, called template.

4.2.1 Model and algorithm

Using the formulation in Allassonnière et al. (2010), let $(y_i)_{1\leq i\leq n}$ be n observed gray level images. Each image y_i is defined on a grid of pixels $\Lambda \subset \mathbb{R}^2$, where for each $s \in \Lambda$, x_s is the location of pixel s in a domain $D \subset \mathbb{R}^2$. We assume that every image derives from the deformation of a common unknown template I, which is a function from D to $\mathbb{R}$. Furthermore, we assume that for every image y_i there exists an unobserved deformation field $\Phi_i : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$y_i(s) = I(x_s - \Phi_i(x_s)) + \sigma\varepsilon_i(s),$$

where $\varepsilon_i(s)$ have standard normal distribution and σ^2 denotes the variance. To formulate this complex problem in a simpler way, the template I and the deformations Φ_i are supposed to have the following parametric form. Given a set of landmarks denoted by $(p_k)_{1\leq k\leq k_p}$, which covers the domain D, the template function I is parametrized by coefficients $\xi \in \mathbb{R}^{k_p}$ through

$$I_\xi = \mathbf{K_p}\xi, \quad \text{where} \quad (\mathbf{K_p}\xi)(x) = \sum_{k=1}^{k_p} \mathbf{K_p}(x, p_k)\xi(k),$$

and $\mathbf{K_p}$ is a fixed known kernel. Likewise, for another fixed set of landmarks $(g_k)_{1\leq k\leq k_g} \in D$, the deformation field is given by

$$\Phi_i(x) = (\mathbf{K_g}z_i)(x) = \sum_{k=1}^{k_g} \mathbf{K_g}(x, g_k)(z_i^{(1)}(k), z_i^{(2)}(k)),$$

where $z_i = (z_i^{(1)}, z_i^{(2)}) \in \mathbb{R}^{k_g} \times \mathbb{R}^{k_g}$ and again, $\mathbf{K_g}$ is a fixed known kernel. The latent variables z_i are centered Gaussian variables with covariance matrix Γ. We refer to Allassonnière et al. (2010) for further details on the model and also for the implementation of the MCMC-SAEM algorithm, which estimates all model parameters (ξ, Γ, σ^2), and so the template I.

4.2.2 Numerical results

In our numerical study we compare the performance of the standard MCMC-SAEM algorithm to the mini-batch version on images from the United States Postal Service database (Hull, 1994).

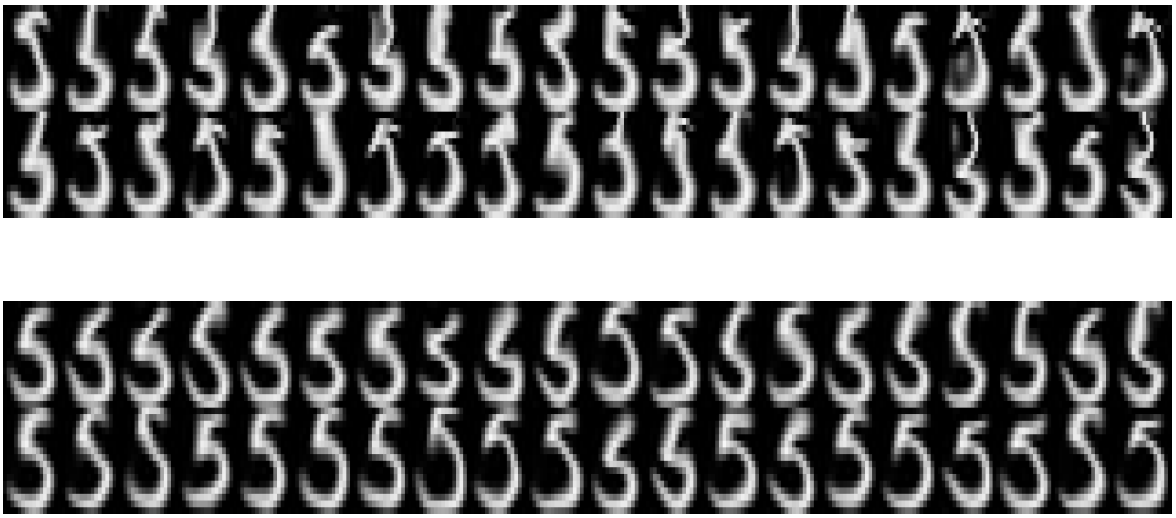

Fig. 4 Synthetic images sampled from the model for digit 5 using the parameter estimates obtained with the batch version on 20 images (top) and with the mini-batch version with $\alpha = 0.2$ on 100 images (bottom).

In the first experiment we set the mini-batch proportion α to 0.1 and have a look on the performances during the very first iterations of the algorithms. Figure 3 shows the estimated template for digit 5 after the first three epochs, that is, after three passes through the dataset. We observe that the mini-batch algorithm obtains a more contrasted and accurate template estimate than the batch version. That is, convergence is accelerated at the beginning of the algorithm when mini-batch sampling is used. This is very similar to our observations in the nonlinear mixed model in the previous section.

Now the question is how to take advantage of this speed up in practice, as we usually do not stop any algorithm after only three epochs. As it is good use to run any MCMC-SAEM algorithm until convergence, our approach is the following. Suppose that we find the computing time acceptable when running the batch algorithm on $n = 20$ images during 1000 iterations, that is, until convergence. Can we do something better by using mini-batch sampling, say with $\alpha = 0.2$, within the same computing time? When applying the mini-batch algorithm on the same 20 images, convergence is attained much faster and there is a gain in computing time, but probably also some loss in accuracy. This is not what we are interested in, as we want to make use of the entire allotted computing time. The solution is to increase the number of images in the input of the mini-batch algorithm. Our reasoning is the following: in the batch version 20 images are processed per iteration. So the mini-batch algorithm with $\alpha = 0.2$ can be applied to $n = 100$ images, as in average only 20 images are used per iteration. Hence, running the mini-batch algorithm with $\alpha = 0.2$ and $n = 100$ and the batch version with $n = 20$ over the same number of iterations takes almost exactly the same time. We see that, given a constraint on the computing time, using a small mini-batch proportion allows to increase the number of images in the input.

To assess the accuracy of the estimates obtained by the two algorithms, we generate new samples from the model using the parameter estimates obtained by the different algorithms. From the synthetic images presented in Figure 4 we see that the ones in the lower part of the figure resemble more usual handwritten digits 5 than the ones in the upper part. This highlights that both template and deformation are better estimated by the mini-batch version performed on 100 images of the dataset than with the batch algorithm on 20 images. Hence, given a constraint on the computing time, more accuracy can be obtained by using the mini-batch MCMC-SAEM instead of the original algorithm.

4.3 Stochastic block model

Now we turn to a random graph model which is interesting as it has a complex dependency structure. The model is the so-called stochastic block model (see Matias and Robin (2014) for a review), where every observation depends on more than one latent component.

4.3.1 Model

In the stochastic block model (SBM) the latent variable $\mathbf{z} = (z_1, \ldots, z_n)$ is composed of i.i.d. random variables

z_i taking their values in $\{1,\dots,Q\}$ with probabilities $p_q = \mathbb{P}_\theta(z_1 = q)$ for $q = 1,\dots,Q$. The observed adjacency matrix $\mathbf{y} = (y_{i,j})_{i,j}$ of a directed graph is such that the observations $y_{i,j}$ are independent conditional on $\mathbf{z}$ and $y_{i,j}|\mathbf{z}$ has Bernoulli distribution with parameter ν_{z_i,z_j} depending on the latent variables of the interacting nodes i and j.

We see that every observation $y_{i,j}$ depends on two latent components, namely on z_i and z_j. In turn, the latent component z_k influences all observations in the set $\{y_{i,k}, i = 1,\dots,n\} \cup \{y_{k,j}, j = 1,\dots,n\}$ creating complex stochastic dependencies between the observations.

Denote $\theta = ((p_q)_{1\le q\le Q}, (\nu_{q,l})_{1\le q,l\le Q})$ the collection of model parameters for a directed SBM. The complete log-likelihood function is given by

$$\begin{aligned}\log \mathbb{P}_\theta(\mathbf{y},\mathbf{z}) &= \log \mathbb{P}_\theta(\mathbf{y}|\mathbf{z}) + \log \mathbb{P}_\theta(\ \mathbf{z})\\ &= \sum_{q=1}^{Q}\sum_{i=1}^{n} \mathbf{1}\{z_i = q\}\log p_q\\ &\quad + \sum_{q,l}\sum_{i,j} \mathbf{1}\{z_i = q, z_j = l\} y_{i,j}\log \nu_{q,l}\\ &\quad + \sum_{q,l}\sum_{i,j} \mathbf{1}\{z_i = q, z_j = l\}(1-y_{i,j})\log(1-\nu_{q,l})\end{aligned}$$

where $\mathbf{1}\{A\}$ denotes the indicator function of the set A. Hence, the complete log-likelihood of the SBM belongs to the exponential family with the following sufficient statistics, for $1 \le q,l \le Q$,

$$\begin{aligned}S_1^q(\mathbf{z}) &= \sum_{i=1}^{n} \mathbf{1}\{z_i = q\},\\ S_2^{q,l}(\mathbf{z}) &= \sum_{i,j} \mathbf{1}\{z_i = q, z_j = l\} y_{i,j},\\ S_3^{q,l}(\mathbf{z}) &= \sum_{i,j} \mathbf{1}\{z_i = q, z_j = l\}(1-y_{i,j}).\end{aligned}$$

and corresponding natural parameters $\varphi_1^q(\theta) = \log p_q$, $\varphi_2^{q,l}(\theta) = \log \nu_{q,l}$ and $\varphi_3^{q,l}(\theta) = \log(1-\nu_{q,l})$.

4.3.2 Algorithm

The implementation is straightforward. In the simulation step, the proposal distribution q of latent variables in the Metropolis algorithm is the discrete uniform distribution on $\{1,\dots,Q\}$.

As in other models, the update of the sufficient statistic $S(\mathbf{z})$ can be numerically optimized. Indeed, with $\mathcal{I}_k$ denoting the indices of latent components that are simulated, the vectors $\mathbf{z}_k$ and $\mathbf{z}_{k-1}$ are only different in the components with indices belonging to $\mathcal{I}_k$. As a consequence, the statistic $S_1^q(\mathbf{z}_k)$, for instance, is more rapidly updated by computing

$$S_1^q(\mathbf{z}_{k-1}) + \sum_{i\in\mathcal{I}_k} (\mathbf{1}\{z_{k,i} = q\} - \mathbf{1}\{z_{k-1,i} = q\})$$

than by the formula $\sum_{i=1}^{n} \mathbf{1}\{z_{k,i} = q\}$, which involves more operations. Likewise, $S_2^{q,l}(\mathbf{z}_k)$ is faster computed as follows

$$\begin{aligned}S_2^{q,l}(\mathbf{z}_k) &= S_2^{q,l}(\mathbf{z}_{k-1}) + \sum_{i\in\mathcal{I}_k}\sum_{j=1}^{n} \mathbf{1}\{z_{k,i} = q, z_{k,j} = l\} y_{i,j}\\ &\quad - \sum_{i\in\mathcal{I}_k}\sum_{j=1}^{n} \mathbf{1}\{z_{k-1,i} = q, z_{k-1,j} = l\} y_{i,j}\\ &\quad + \sum_{i=1}^{n}\sum_{j\in\mathcal{I}_k} \mathbf{1}\{z_{k,i} = q, z_{k,j} = l\} y_{i,j}\\ &\quad - \sum_{i=1}^{n}\sum_{j\in\mathcal{I}_k} \mathbf{1}\{z_{k-1,i} = q, z_{k-1,j} = l\} y_{i,j}.\end{aligned}$$

Here we see that not the entire data $\mathbf{y}$ are visited to update $S_2^{q,l}$, but only those observations $y_{i,j}$ that stochastically depend on the updated latent components z_i with $i \in \mathcal{I}_k$.

The maximisation of the complete likelihood function with given values for the sufficient statistics $s_1^q, s_2^{q,l}$ and $s_3^{q,l}$ is straightforward. The updated parameter values are given by $p_q = s_1^q/n$ and $\nu_{q,l} = s_2^{q,l}/(s_2^{q,l} + s_3^{q,l})$.

4.3.3 Simulation results

For our simulations, model parameters are set to $\pi_1 = 1 - \pi_2 = 0.6$ and

$$(\nu_{q,l})_{q,l} = \begin{pmatrix} 0.25 & 0.1 \\ 0.1 & 0.2 \end{pmatrix}.$$

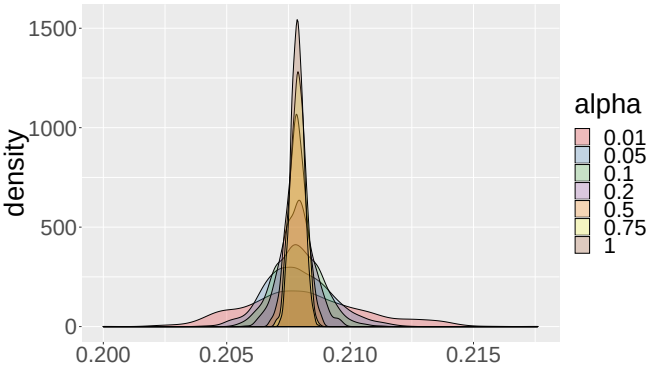

Fig. 5 Estimation of the limit distribution of the estimate of $\nu_{2,2} = 0.2$ after 10 000 iterations for the mini-batch algorithm with proportions $\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.5, 0.75, 1\}$.

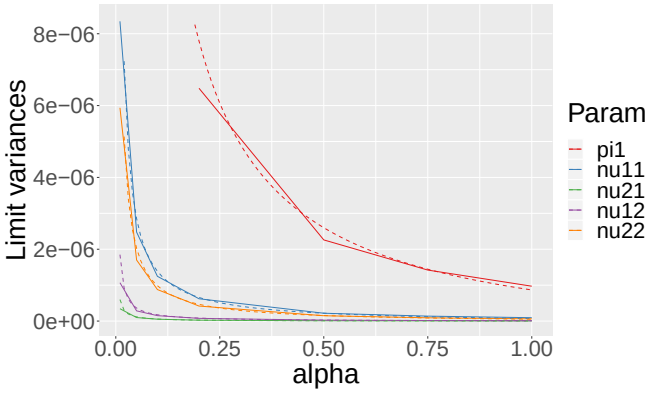

Fig. 6 Sample variances of the parameter estimates after 10 000 iterations as a function of the mini-batch proportion α (solid lines) and adjusted theoretical limit variances (dashed lines).

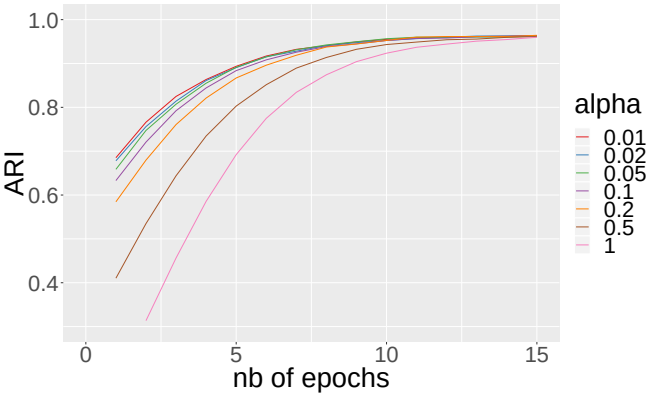

Fig. 7 Mean ARI obtained by mini-batch MCMC-SAEM algorithms as a function of the number of epochs for $\alpha \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$.

To study the impact of mini-batch sampling on the asymptotic behavior of the estimates, a directed graph with $n = 100$ nodes is generated from the model and the mini-batch algorithm with α ranging from 0.01 to 1 and with 10 000 iterations is applied 1000 times.

Figure 5 shows the histograms of the estimates of parameter $\nu_{2,2} = 0.2$ obtained with the different algorithms. All histograms are approximately unimodal, centered at the same value and symmetric. Moreover, we see that the larger the mini-batch size the tighter the distribution. Indeed, it seems that the estimates are asymptotically normal and the limit variance increases when α decreases. This increase of the limit variance induced by mini-batch sampling is illustrated for all model parameter estimates in Figure 6. Furthermore, Figure 6 checks whether the theoretical formula of the limit variance derived in Section 3.3 is adequate. Recall that we conjecture that the limit variance obtained with the mini-batch algorithm with proportion α equals $(2-\alpha)/\alpha$ times the limit variance obtained with the batch algorithm, here represented by the dashed lines. The excellent fit supports our conjecture.

Concerning the behavior at the beginning of the algorithm, we observe the same acceleration for the mini-batch versions as in the other models. Here 500 datasets are simulated from the SBM and Figure 7 shows the evolution of the adjusted rand index (ARI) during the first epochs (Hubert and Arabie, 1985). This index compares the clustering of the nodes obtained by the algorithm to the true block memberships given by the latent components z_i. The ARI equals one if two clusterings are identical up to permutation of the block labels. We see that the algorithm with the smallest mini-batch proportion provides the best clustering at any given number of epochs. Finding the good clustering is essential for accurate parameter estimates.

Finally, we have a closer look on computing time aspects. As already mentioned, the computing time of the M-step does not depend on the mini-batch proportion α. It is also clear that the simulation step in the mini-batch algorithm is in average α times the computing time of the simulation step in the batch version, as only a proportion α of the latent components are simulated. However, the computing time of the stochastic approximation step, and in particular the update of the sufficient statistic, depends on the amount of data used and thus on the dependency structure of the model. In most models, where every observation only depends on a single latent component z_i, the proportion of data involved in the update is α. However, in the SBM a set of latent components $\{z_1, \ldots, z_m\}$ for $m < n$ influences the set of observations $\{y_{i,j}, i = 1, \ldots, m, j = 1, \ldots, n\} \cup \{y_{i,j}, i = 1, \ldots, n, j = 1, \ldots, m\}$, whose cardinality is $2mn - m^2$. It follows that for a mini-batch proportion α the corresponding proportion of data used to update the sufficient statistics is $\alpha(2-\alpha)$ and so the computing time of this step is $\alpha(2-\alpha)$ times the computing time of the stochastic approximation step in the batch algorithm.

Let us call SAE-step the combination of the simulation step and the stochastic approximation step. We determine the median computing time of the SAE-step over 100 runs of the SAE-step for different mini-batch proportions α and different numbers of nodes n. Figure 8 shows the ratio of the median time of the mini-batch SAE-step with proportion α over the median time of the batch SAE-step for different numbers of nodes, that is,

$$\alpha \mapsto \frac{\text{median time of SAE-step with } \alpha}{\text{median time of batch SAE-step}}.$$

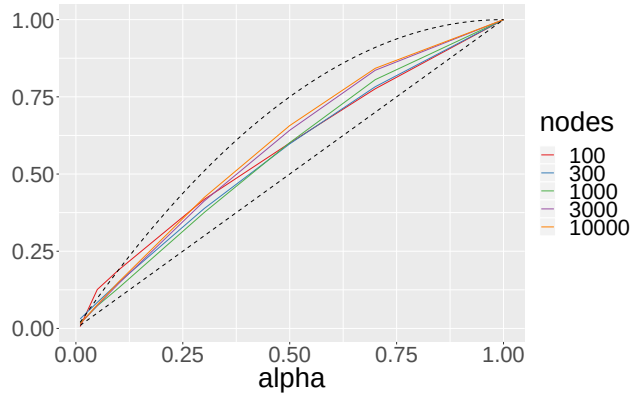

Fig. 8 Ratio of the median computing time of the mini-batch SAE-step with α over the median time of the batch SAE-step for different numbers of nodes (solide lines). The dashed lines represent the functions $\alpha \mapsto \alpha$ and $\alpha \mapsto \alpha(2-\alpha)$, the first corresponds to the expected ratio of computing times of the simulation step, the latter to the expected ratio of of the stochastic approximation step.

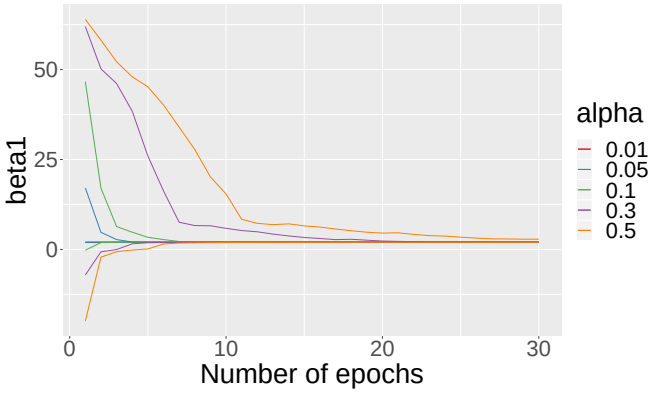

Fig. 9 Evolution of 80%-confidence bands for β_1 with respect to the number of epochs for mini-batch proportions $\alpha \in \{0.01, 0.05, 0.1, 0.3, 0.5, 0.8, 1\}$

The dashed lines represent the functions $\alpha \mapsto \alpha$ and $\alpha \mapsto \alpha(2-\alpha)$. The first corresponds to the expected ratio of computing times of the simulation step, the latter to the expected ratio of computing times of the stochastic approximation step. As expected, the observed computing time ratios of the entire SAE-step fall between these two boundaries.

4.4 Frailty model in survival analysis

In survival analysis the frailty model is an extension of the well-known Cox model (Duchateau and Janssen, 2008). The hazard rate function in the frailty model includes an additional random effect, called frailty, to account for unexplained heterogeneity.

4.4.1 Model and algorithm

The observations are survival times $\mathbf{t} = (t_{ij})_{1\le i\le n, 1\le j\le m}$ measured over n groups with m measurements per group, and covariates $\mathbf{X}_{ij} \in \mathbb{R}^p$. We denote by λ_0 the baseline hazard function, that is here chosen to be the Weibull function given by

$$\lambda_0(t) = \lambda_0 \rho t^{\rho-1}, \quad t > 0,$$

with $\lambda_0 > 0$ and $\rho > 1$. For every group i, a latent variable z_i is introduced representing a frailty term. We suppose that $z_1, \dots, z_n$ are i.i.d. with centered Gaussian distribution with variance σ^2. The conditional hazard rate $\lambda_{ij}(\cdot|z_i)$ of observation t_{ij} given the frailty z_i is defined as

$$\lambda_{ij}(\cdot|z_i) = \lambda_0(\cdot)\exp(\mathbf{X}_{ij}{}^\intercal \boldsymbol{\beta} + z_i),$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$. Thus, the unknown model parameter is $\theta = (\boldsymbol{\beta}, \sigma^2, \lambda_0, \rho)$. In practical applications the main interest lies in the estimation of the regression parameter $\boldsymbol{\beta}$.

In the frailty model the conditional survival function is given by

$$\begin{aligned} G_{ij}(t|z_i) &= \mathbb{P}(t_{ij} > t|z_i) \\ &= \exp[-\lambda_0 t^\rho \exp(\mathbf{X}_{ij}{}^\intercal \boldsymbol{\beta} + z_i)]. \end{aligned}$$

In other words, the conditional distribution of the survival time t_{ij} given z_i is the Weibull distribution with scale parameter $\lambda_0 \exp(\mathbf{X}_{ij}{}^\intercal \boldsymbol{\beta} + z_i)$ and shape parameter ρ. For the conditional and the complete likelihood functions we obtain

$$\mathbb{P}_\theta(\mathbf{t}|\mathbf{z}) = \prod_{i=1}^n \prod_{j=1}^m G_{ij}(t_{ij}|z_i)\lambda_{ij}(t_{ij}|z_i),$$

$$\mathbb{P}_\theta(\mathbf{t}, \mathbf{z}) = \mathbb{P}_\theta(\mathbf{t}|\mathbf{z}) \prod_{i=1}^n \varphi\left(\frac{z_i}{\sigma}\right),$$

where φ denotes the density of the standard normal distribution.

The implementation of the algorithm is straightforward. In the simulation step candidates $\tilde{z}_{k,i}$ are drawn from the normal distribution $\mathcal{N}(z_{k-1,i}, 0.2)$. In the M-step, the updates of σ^2 and $\lambda_{0,k}$ are explicit, while those of $\boldsymbol{\beta}$ and ρ are obtained by the Newton-Raphson method.

4.4.2 Numerical results

In a simulation study we consider the frailty model with parameters fixed to $\boldsymbol{\beta} = (\beta_1, \beta_2) = (2, 3), \lambda_0 = 3$, $\sigma^2 = 2$ and $\rho = 3.6$. We set $n = 5000$ and $m = 100$. The covariates $\mathbf{X}_{ij}$ are drawn independently from the uniform distribution for every dataset.

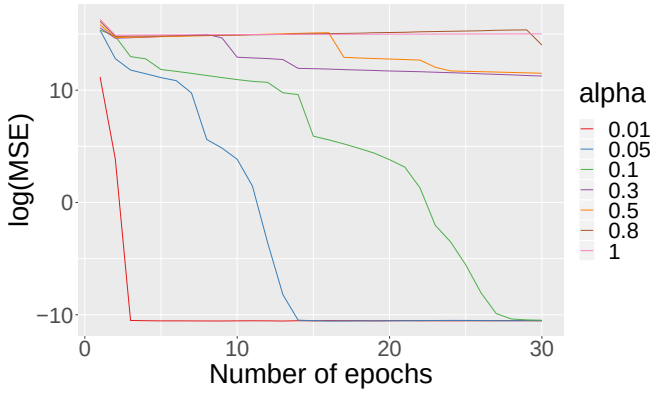

Fig. 10 Evolution of the logarithm of the empirical mean squared error of estimates of β_1 with respect to the number of epochs for mini-batch proportions $\alpha \in \{0.01, 0.05, 0.1, 0.3, 0.5, 0.8, 1\}$.

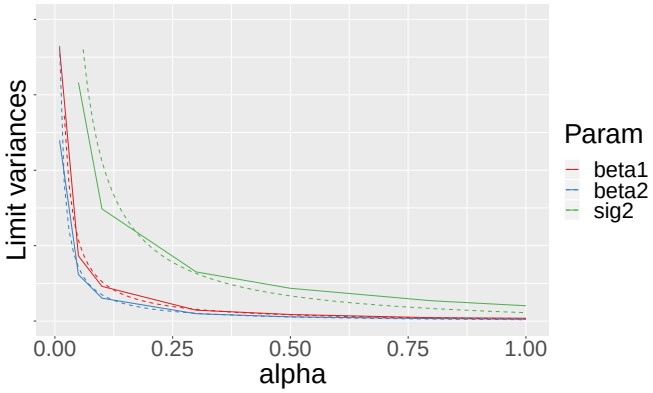

Fig. 12 Sample variances of the parameter estimates after 8000 iterations as a function of the mini-batch proportion α (solid lines) and adjusted theoretical limit variances (dashed lines).

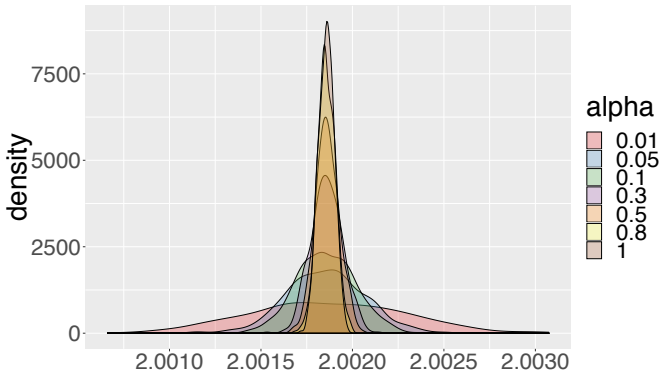

Fig. 11 Estimation of the limit distribution of the estimate of β_1 after 8000 iterations for the mini-batch algorithm with proportions $\alpha \in \{0.01, 0.05, 0.1, 0.3, 0.5, 0.8, 1\}$.

In the first setting, 500 datasets are generated and the mini-batch MCMC-SAEM algorithm with random initial values and mini-batch proportions α between 0.01 and 1 is applied. Figure 9 and 10 shows the evolution of the precision of the mean of the estimates $\bar{\beta}_{1,k} = \sum_{l=1}^{k} \beta_{1,l}/k$ of parameter component β_1 as a function of the number of epochs. Figure 9 shows 80%-confidence bands for β_1, and Figure 10 gives the corresponding evolution of the logarithm of the empirical mean squared error. Again, it is clear from both graphs that the rate of convergence depends on the mini-batch proportion. At (almost) any number of epochs, the mean squared error and the width of the confidence intervals is increasing in α. From Figure 9 and 10 we see, for instance, that the mini-batch version with $\alpha = 0.01$ attains convergence after only three epochs, while almost 30 epochs are required when $\alpha = 0.1$. The choice of α that achieves the fastest convergence is the smallest mini-batch proportion, here 0.01.

In the second setting, we study the asymptotic behavior of the estimates, when the algorithms are supposed to have converged, that is after 8000 iterations. To evaluate the variance of the estimates that is only due to the stochasticity of the algorithm, we fix a dataset and run the different algorithms 500 times using different initial values. Figure 11 illustrates the limit distributions of the estimates of β_1, which seem to be Gaussian, centered at the same value, but with varying variances. Figure 12 gives the corresponding values of the limit variances for the different parameter estimates. Again, we see that the limit variance increases when α decreases. This is expected as less data are visited per iteration for smaller α. Furthermore, we conjecture in Section 3.3 that the theoretical limit variance of the sequence generated by the algorithm with minibatch size α is equal to $V_1(2-\alpha)/\alpha$ where V_1 stands for the limit variance of the batch algorithm. Figure 12 shows a very good fit of the sample variances of the different parameters to the function $\alpha \mapsto v_1(2-\alpha)/\alpha$.

5 Conclusion

In this paper we have proposed to perform mini-batch sampling in the MCMC-SAEM algorithm. We have shown that the mini-batch algorithm belongs to the family of classical MCMC-SAEM algorithms with specific transition kernels. It is also shown that the mini-batch algorithm converges almost surely, whenever the original algorithm does. Concerning the limit distribution, according to heuristic arguments and simulation results in different models, estimators are asymptotically normal and we have quantified the impact of the mini-batch proportion onto the limit variance. However, the formal proof of this result is left for future work.

The numerical experiments carried out in various latent variable models illustrate that mini-batch sampling leads to an important speed-up of convergence at the beginning of the algorithm compared to the original algorithm. In most papers on mini-batch sampling as well as in this paper, the illustration of this speed up is based on a comparison of estimates at the same number of epochs. However, the computing time of an epoch depends on the mini-batch proportion α, on the amount of data used in the stochastic approximation step and on the computing time of the M-step, which is performed much more often when α is small. Indeed, the smaller the value of α, the larger the number of M-steps within an epoch. In the frailty model, for instance, the maximisation in the M-step is not explicit and thus more time-consuming than the other steps of the algorithm, which makes mini-batch sampling less attractive for practical use in this model. This also raises the question of the interest of an analysis of the performance of the estimators with respect to the number of epochs. From a practical point of view, in future studies, we advocate that it would be much more appealing to compare algorithms relying on the same computing time rather than on the same number of epochs.

The study of the stochastic block model has shown that the common presentation of mini-batch sampling as a method where a subset of the data is selected to perform an iteration is misleading. Indeed, in the algorithm, first the latent components that are to be simulated are chosen, and only then, the data that are associated with these updated latent components are determined to perform the update of the sufficient statistic. In models where every observation only depends on a single latent component, the proportion of data used for the update equals the proportion of simulated latent components. However, in models with a more involved dependency structure as SBM this does no longer hold. As a consequence, in the SBM the computing time of one iteration of the SAE-step in the mini-batch algorithm is not α times the corresponding SAE-step in the batch version.

These issues on the computing time lead us to the important question of how to make good use of mini-batch sampling in practice, where we are often confronted to constraints on the computing time. It seems to us that the relevant problem is to find the algorithm that provides the best results within some allotted computing time. As we have seen in Section 4.2, combining mini-batch sampling with an increase of the number of observations compared to the batch version is a means to achieve more accurate estimates under a given constraint on the computing time. Indeed, increasing the sample size for mini-batch algorithms compensates for the loss in accuracy of the final estimates, while the acceleration of the convergence at the beginning of the algorithm ensures the convergence of the MCMC-SAEM algorithm within the considered computing time.

To find the optimal mini-batch proportion α and the associated optimal sample size, an analysis of the computing time per iteration is required, instead of per number of epochs. These optimal values are model dependent. Furthermore, as any MCMC-SAEM algorithm must always be run until convergence, it is necessary to understand the impact of the mini-batch proportion α and the sample size on the convergence of the algorithm, that is, on the number of iterations required to achieve convergence.

All programs are available on request from the authors.

Acknowledgements Work partly supported by the grant ANR-18-CE02-0010 of the French National Research Agency ANR (project EcoNet).

References

Allassonnière S, Amit Y, Trouvé A (2007) Toward a coherent statistical framework for dense deformable template estimation. J Roy Statist Soc Ser B 69:3–29

Allassonnière S, Kuhn E, Trouvé A (2010) Construction of Bayesian deformable models via a stochastic approximation algorithm: A convergence study. Bernoulli 16(3):641–678

Andrieu C, Moulines E, Priouret P (2005) Stability of stochastic approximation under verifiable conditions. SIAM J Control Optim 44(1):283–312

Cappé O (2011) Online EM algorithm for hidden Markov models. Journal Computational and Graphical Statistics 20(3):728–749

Cappé O, Moulines E (2009) On-line expectation-maximization algorithm for latent data models. J Roy Statist Soc Ser B 71(3):593–613

Davidian M, Giltinan DM (1995) Nonlinear Models for Repeated Measurement Data. Chapman & Hall/CRC Press

Delyon B, Lavielle M, Moulines E (1999) Convergence of a stochastic approximation version of the EM algorithm. Ann Statist 27(1):94–128

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J Roy Statist Soc Ser B 39(1):1–38

Duchateau L, Janssen P (2008) The frailty model. New York, NY: Springer

Fort G, Moulines E, Roberts GO, Rosenthal JS (2003) On the geometric ergodicity of hybrid samplers. Journal of Applied Probability 40:123–146

Fort G, Jourdain B, Kuhn E, Lelièvre T, Stoltz G (2015) Convergence of the Wang-Landau algorithm. Mathematics of Computation 84(295):2297–2327

Hubert L, Arabie P (1985) Comparing partitions. J Classif 2:193–218

Hull JJ (1994) A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(5):550–554

Karimi B, Lavielle M, Moulines E (2018) On the convergence properties of the mini-batch EM and MCEM algorithms, unpublished

Kuhn E, Lavielle M (2004) Coupling a stochastic approximation version of EM with an MCMC procedure. ESAIM: P&S 8:115–131

Kuhn E, Lavielle M (2005) Maximum likelihood estimation in nonlinear mixed effects models. Comput Stat Data An 49(4):1020–1038

Lange K (1995) A gradient algorithm locally equivalent to the EM algorithm. J Roy Statist Soc Ser B 2(57):425–437

Liang P, Klein D (2009) Online EM for unsupervised models. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, NAACL '09, pp 611–619

Matias C, Robin S (2014) Modeling heterogeneity in random graphs through latent space models: a selective review. ESAIM: Proceedings and Surveys 47:55–74

Neal RM, Hinton GE (1999) A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Jordan MI (ed) Learning in Graphical Models, MIT Press, Cambridge, MA, USA, pp 355–368

Nguyen H, Forbes F, McLachlan G (2019) Mini-batch learning of exponential family finite mixture models. Tech. rep., arXiv:1902.03335

Robert CP, Casella G (2004) Monte Carlo statistical methods, 2nd edn. Springer Texts in Statistics, Springer-Verlag, New York

Titterington DM (1984) Recursive parameter estimation using incomplete data. J Roy Statist Soc Ser B 2(46):257–267