Online EM for Functional Data

Florian Maire, Éric Moulines, Sidonie Lefebvre

Online EM for functional data

Florian Maire [a], Eric Moulines [b], Sidonie Lefebvre [c]

[a] *School of Mathematics and Statistics, University College Dublin, Ireland*
[b] *CMAP, École Polytechnique, 91128 Palaiseau, France*
[c] *ONERA - The French Aerospace Lab, F-91761 Palaiseau, France*

A novel approach to perform unsupervised sequential learning for functional data is proposed. The goal is to extract reference shapes (referred to as *templates*) from noisy, deformed and censored realizations of curves and images. The proposed model generalizes the Bayesian dense deformable template model, a hierarchical model in which the template is the function to be estimated and the deformation is a nuisance, assumed to be random with a known prior distribution. The templates are estimated using a Monte Carlo version of the online Expectation–Maximization (EM) algorithm. The designed sequential inference framework is significantly more computationally efficient than equivalent batch learning algorithms, especially when the missing data is high-dimensional. Some numerical illustrations on curve registration problem and templates extraction from images are provided to support the methodology.

Keywords:
Online Expectation–Maximization algorithm
Deformable templates models
Unsupervised clustering
Markov chain Monte Carlo
Carlin and Chib algorithm
Big Data

1. Introduction

Functional data analysis is concerned with the analysis of curves and shapes, which often display common patterns but also variations (in amplitude, orientations, time–space warping, etc…). Extracting common patterns (referred to as *templates*) from functional data, and the related problem of curves/images registration has given raised to a wealth of research efforts; see Ramsay (2006), Zhong (2008), Ramsay (2011) and the references therein and Wu and Hitchcock (2016) and Nguyen et al. (2016) for two recent contributions.

Most of the proposed techniques used so far have been developed in a supervised classification context. The method typically aims at finding a time/space warping transformation allowing to synchronize/register all the observations associated to a given class of curves/shapes and to estimate a template by computing a cross-sectional mean of the aligned patterns. In most cases, the deformation is penalized, to favor "small" time/space shifts. Many different deformation models

have been proposed for curves and for images. For curves, the warping function is often assumed to be monotone increasing. In this context, the dynamic time warping algorithm is by far the most popular algorithm: it enables the alignment of curves by minimizing a cost function of the warping path, which can be solved by a dynamic programming algorithm (Wang and Gasser, 1997). Non-parametric (Kneip and Gasser, 1992; Silverman, 1985; Ramsay and Li, 1998) as well as Bayesian approaches (Telesca and Inoue, 2008; Liu and Yang, 2009; Wu and Hitchcock, 2016) have also been proposed, but they are still far less popular. The situation is more complex for shapes and images. Different deformation models have been proposed, involving rigid deformations, small deformations (Castellanos et al., 2004) or deformation fields ruled by a differential equation; see Christensen (1999).

In this paper, we introduce a common Bayesian statistical framework for *unsupervised* clustering and template extraction, with applications to curve synchronization and shape registration. Following the seminal work by Allassonnière and Kuhn (2010) and Allassonnière et al. (2007), we generalize the mixture of *deformable template models*. This approach models an observed curve/shape as a template (defined as a function of time or space), selected from a collection of templates, which undergoes a random deformation and is observed in presence of an additive noise; see Allassonnière et al. (2007), Bigot and Charlier (2011), Christensen et al. (1996) and Allassonnière et al. (2013) for a complete survey. Contrary to the classical time warping/spatial registration algorithms which consist in synchronizing all the observations of a shape in a supervised framework, the mixture of deformable template models is an unsupervised classifier: it estimates functional templates from a set of shapes/curves and considers the time warping/spatial deformations as random nuisance parameters. It is important to stress that the model allows the integration of the deformation conditional on the observations while considering the templates as unknown deterministic functional parameters. In this context, the deformation might be seen as a *random effect*, which is similar to random effects in linear mixed models in longitudinal data analysis. Whereas this change in perspective might seem rather benign, it makes a huge difference both in theory and in practice.

In our model, the warping/deformation function and the cluster index are modeled as hidden data and we consequently turn to an Expectation–Maximization (EM)-type algorithm (Dempster et al., 1977) to estimate the templates. However, in our model the conditional expectation of the complete data log-likelihood is analytically intractable, compromising a plain EM implementation. This situation has raised a significant research interest over the last decades and several versions of so-called stochastic EM, in which the E-step is approximated, have been successfully applied to the template extraction problem. A rough approximation of the conditional expectation was considered in Ma et al. (2008), in which the posterior distribution is replaced by a point mass located at the posterior mode. Another elementary approach consists in linearizing the deformed template in the neighborhood of its nominal shape, under the assumption of small deformations. This alternative has been considered, among others by Liu and Yang (2009) and Frey and Jojic (2003), in which the transformed mixture of Gaussian models was used. Another way to handle the E-step, suggested in Gaffney and Smyth (2004) and Bernhardt et al. (2015), consists in performing an approximate Bayesian integration, which amounts to replace the posterior distribution of the hidden data conditionally to the observation by a Gaussian distribution, obtained from a Laplace approximation. Here again, such approximations are difficult to justify in our context. The expectation can also be approximated by Markov chain Monte Carlo, an idea which was put forward by Allassonnière and Kuhn (2010) and Kuhn and Lavielle (2004), extending the original Stochastic Approximation EM (SAEM) (Delyon et al., 1999) and known as the MCMC-SAEM algorithm. This algorithm has been theoretically justified (Kuhn and Lavielle, 2004) and has shown to perform satisfactorily in the template extraction application (Allassonnière and Kuhn, 2010). However it turns out to be a time-consuming solution especially when a large number of observations are available and the dimension of the missing data is huge. The extension of the model to multiple classes is even more computationally involved. This concern calls for more sophisticated and efficient MCMC samplers, based for example on Langevin dynamics as successfully considered in Allassonniere and Kuhn (2015). Finally, in Moffa and Kuipers (2014), the authors argue that integrating the E-step with a Sequential Monte Carlo mechanism is more adapted to the EM iterative structure, but degeneracy of particles remains a potential issue, especially in mixture models.

We propose the Monte Carlo online EM (MCoEM), an online algorithm in which the curves/shapes are processed one at a time and only once, allowing to estimate the unknown parameters of the mixture of deformable templates model. We adapt the online EM algorithm proposed in Cappé and Moulines (2009) to intractable E-step settings whereby casting MCoEM as a *noisy* online EM. Our model is too general to allow the linearization or the use of Gaussian approximation of the complete data log-likelihood, as it was done in Liu and Yang (2009) and Gaffney and Smyth (2004). We thus propose to approximate the conditional expectation thanks to an MCMC algorithm adapted from the celebrated Carlin and Chib algorithm (Carlin and Chib, 1995). Indeed, working online implies processing the data on the fly without storing them afterwards and this requires the posterior distribution exploration to be more accurate than in the MCMC-SAEM framework which refines the state-space exploration gradually, at each EM iteration. Building an online learning framework for template extraction has a two-fold motivation: (i) the data need not be stored which can be useful should the algorithm be implemented on a portable device with limited memory/energy resources and (ii) MCoEM reduces significantly the computational burden that would be generated by an equivalent batch algorithm such as the MCMC-SAEM (Kuhn and Lavielle, 2004).

This paper is organized as follows: in Section 2 the mixture of the dense deformable template model is generalized and the Monte Carlo online EM algorithm is presented in Section 3. The sampling method of the posterior distribution, necessary to approximate the E-step, is proposed in Section 4. Illustrations of templates obtained by applying MCoEM to curves and shapes are proposed in Section 5 and are compared with those obtained using MCMC-SAEM. An application of the methodology to a classification problem is provided in Section 6 and shows how competitive MCoEM is over batch

equivalent algorithms. Benefits and shortcomings of the MCoEM methodology are discussed in Section 7 and perspectives are raised.

2. A mixture of deformable template models

2.1. A basic deformable model

In this section, we introduce a basic model for curves and images. A *template* is a function defined on a space $\mathbb{U}$ and taking for simplicity real values. Typically, for curves $\mathbb{U} = \mathbb{R}$ and for shapes $\mathbb{U} = \mathbb{R}^2$. We denote by $\mathbb{F}$ the set of templates.

The observations are modeled as the stochastic process Y indexed by $u \in \mathbb{U}$ and given by:

$$Y(u) = \lambda f \circ D(u, \beta) + \sigma W(u), \tag{1}$$

where $f \in \mathbb{F}$ is a template function, $\lambda \in \mathbb{R}^{+*}$ is a scaling factor, $\sigma^2 \in \mathbb{R}^{+*}$ is the noise variance and W a Gaussian process with zero-mean, unit variance and known covariance function. D is a function, belonging to $\mathbb{D}$, the set of mappings from $\mathbb{U}$ to itself and parameterized by a vector $\beta \in \mathbb{B}$, where $\mathbb{B}$ is an open subset of some Euclidean space of dimension d_β. For curves, $\mathbb{D}$ can be chosen as the homotheties and translations mappings and more generally as the set of monotone functions (with appropriate smoothness conditions). For shapes, $\mathbb{D}$ can be taken as the set of rigid transformations of the plane, such as rotations, homotheties or translations along with a local deformation field. The models for the set of deformations $\mathbb{D}$ are problem dependent; see Section 5.

In this setting, β and λ are random variables and each realization of Y follows from different realizations of β and λ. The quantity of interest is the template f (a deterministic functional parameter), while the deformation D and the global scaling λ are regarded as nuisance parameters, that should be integrated out.

Finally, we assume that the set of templates $\mathbb{F}$ is the linear subspace spanned by the basis vectors $\{\phi_\ell\}_{1 \le \ell \le m}$. Hence, a template $f_{\boldsymbol{\alpha}} \in \mathbb{F}$ may be expressed as:

$$f_{\boldsymbol{\alpha}} = \sum_{\ell=1}^{m} \alpha_\ell \phi_\ell, \quad \text{where } \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^T \in \mathcal{A}, \tag{2}$$

where for all $\ell \in \{1, \ldots, m\}$, $\phi_\ell : \mathbb{U} \to \mathbb{R}$ and $\mathcal{A}$ is a subset of $\mathbb{R}^m$. The pattern is observed at some design points denoted $\Omega = \{u_1, \ldots, u_{|\Omega|}\}$, where $|\Omega|$ is the dimension of the observations such that for all $s \in \{1, \ldots, |\Omega|\}$, $u_s \in \mathbb{U}$. Let Φ_β be the $|\Omega| \times m$ matrix defined such that for all (s, ℓ) in $\{1, \ldots, |\Omega|\} \times \{1, \ldots, m\}$,

$$[\Phi_\beta]_{s,\ell} = \phi_\ell \circ D(u_s, \beta). \tag{3}$$

Defining $\mathbf{Y} = (Y(u_1), \ldots, Y(u_{|\Omega|}))^T$ and $\mathbf{W} = (W(u_1), \ldots, W(u_{|\Omega|}))^T$ and using (1), the vector of observations can be expressed in a matrix–vector form as:

$$\mathbf{Y} = \lambda \Phi_\beta \boldsymbol{\alpha} + \sigma \mathbf{W}. \tag{4}$$

2.2. A mixture of deformable templates

We extend the model to include multiple templates corresponding to the different "typical" shapes that we are willing to cluster and then recognize. To that purpose, we construct a mixture of the template model introduced in the previous section. Denote by C the number of classes $(\mathcal{C}_1, \ldots, \mathcal{C}_C)$. We associate to each observation $\mathbf{Y}$ a (hidden) class index $I \in \mathbb{I}$, where $\mathbb{I} = \{1, \ldots, C\}$. To each class $\{\mathcal{C}_j\}_{j \in \mathbb{I}}$ is attached a template function $\{f_j\}_{j \in \mathbb{I}}$ in $\mathbb{F}$, which is parameterized by $\{\boldsymbol{\alpha}_j\}_{j \in \mathbb{I}} \in \mathbb{R}^m$. Moreover, a weight $\omega_j \in (0, 1)$ is assigned to the class $I = j \in \mathbb{I}$ and we denote by $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_C)$ the set of prior weights $(\sum_{j=1}^{C} \omega_j = 1)$. To sum up, we consider the following hierarchical model:

$$\mathbf{Y} \in \mathcal{C}_j, \quad \mathbf{Y} = \lambda \Phi_\beta \boldsymbol{\alpha}_j + \sigma \mathbf{W}. \tag{5}$$

It is assumed that the observations $\{\mathbf{Y}_n\}_{n \ge 1}$ are independent random variables, generated as follows:

$$\begin{cases} I_n \sim \text{Multi}(1, \boldsymbol{\omega}), \\ \lambda_n \sim \text{Gamma}(a, 1/a), \\ \beta_n \mid I_n = j \sim \mathcal{N}_{d_\beta}(0_{d_\beta}, \Gamma_j), \end{cases} \tag{6}$$

where Multi denotes the multinomial distribution, a is the shape parameter of the Gamma distribution (assumed to be known). We assume that the scale parameter is $1/a$ to ensure that *a priori* $\mathbb{E}(\lambda_n) = 1$. The choice of a should reflect the expected range of scales, the tolerated variation in scale being inversely proportional to a. Finally, 0_{d_β} is the d_β-dimensional null vector and Γ_j the deformation covariance matrix associated to the class $\mathcal{C}_j$. In Section 5, different covariance models are used in function of the deformation model adopted. We stress that the distribution of the scaling parameter is independent of the class index, while the deformation prior distribution is class-dependent. Indeed, on the one hand, the scaling factor accounts for different ranges of observation and is thus independent of what is actually being observed. On the other hand,

considering different prior distributions for the deformation might help to learn typical relevant distortions for each class and thus ease the warping process.

In the sequel we assume that $\{\mathbf{W}_n\}_{n\geq 1}$ is a vector-valued white noise with zero-mean and identity covariance matrix. The extension to more general covariance is straightforward. Hence, conditionally on the class index I_n, the global scale λ_n and local deformation β_n, the likelihood of $\mathbf{Y}_n$ given the missing data is:

$$\mathbf{Y}_n \mid I_n = j, \lambda_n, \beta_n \sim \mathcal{N}_{|\Omega|}(\lambda_n \Phi_{\beta_n} \boldsymbol{\alpha}_j, \sigma^2 \mathrm{Id}_{|\Omega|}), \tag{7}$$

where $\mathrm{Id}_{|\Omega|}$ is the $|\Omega| \times |\Omega|$ identity matrix. Denote by Θ the set of parameters

$$\Theta = \bigcup_{j=1}^{C} \left\{ \left(\boldsymbol{\alpha}_j, \Gamma_j, \omega_j, \sigma\right) \mid \boldsymbol{\alpha}_j \in \mathcal{A},\ \Gamma_j \in \mathcal{M}^+(\mathbb{R}),\ \omega_j \in (0,1),\ \sigma > 0 \right\} \cap \left\{ \sum_{j=1}^{C} \omega_j = 1 \right\}, \tag{8}$$

where $\mathcal{M}^+(\mathbb{R})$ is the set of $d_\beta \times d_\beta$ positive definite matrices.

Let $\mathbf{X}_n$ be the random vector $\mathbf{X}_n = (\beta_n, \lambda_n)$ taking its values in $\mathbb{X} = \mathbb{B} \times \mathbb{R}^{+*}$ with dimension $d_{\mathbf{X}} = d_\beta + 1$. In the sequel, we will use the formalism and the terminology of the incomplete data model; see McLachlan and Krishnan (2007). In this formalism, the observation $\mathbf{Y}_n$ stands for the incomplete data, $(I_n, \mathbf{X}_n)$ are the missing data and $(I_n, \mathbf{X}_n, \mathbf{Y}_n)$ are the complete data. For a given value of the parameter $\theta \in \Theta$, the complete data likelihood L_θ writes:

$$\mathrm{L}_\theta(I_n, \mathbf{X}_n, \mathbf{Y}_n) = g_\theta(\mathbf{Y}_n \mid I_n, \mathbf{X}_n) p_\theta(\mathbf{X}_n \mid I_n) \omega_{I_n}, \tag{9}$$

where, for a given value of the parameter $\theta \in \Theta$, g_θ is the conditional density of the observations given the missing data and p_θ is prior density of the scaling factor and the local deformation parameter conditionally on the class index. Using (7) and (6), these densities write

$$g_\theta(\mathbf{Y}_n \mid I_n, \mathbf{X}_n) \propto \exp\left(-(1/2\sigma^2)\|\mathbf{Y}_n - \lambda_n \Phi_{\beta_n} \boldsymbol{\alpha}_{I_n}\|^2\right), \tag{10}$$

$$p_\theta(\mathbf{X}_n \mid I_n) \propto \exp\left(-(1/2)\beta_n^T \Gamma_{I_n}^{-1} \beta_n\right) \lambda_n^{a-1} \exp(-a\lambda_n). \tag{11}$$

The incomplete data likelihood is obtained by marginalizing the complete data likelihood with respect to the missing data.

3. Sequential parameter estimation using the Online EM algorithm

In its original version (Dempster et al., 1977), the Expectation–Maximization (EM) is a batch algorithm, *i.e.* that uses a fixed set of observations, performing maximum likelihood estimation in incomplete data models. It produces a sequence of parameters, in such a way that the observed likelihood is increased at each iteration. Each iteration is decomposed into two steps. In the E-step, the conditional expectation of the complete data log-likelihood function given the observations and the current fit of the parameters is computed; in the M-step, the parameters are updated by maximizing the conditional expectation computed in the E-step.

In this paper, we focus on a learning setup in which the observations are obtained sequentially and the parameters are updated as soon as a new observation is available. Among several sequential learning algorithms designed to estimate parameters in missing data models, the online EM algorithm proposed in Cappé and Moulines (2009) (see also Liu et al., 2006 for a similar version in a specific setting) sticks closely to the original EM methodology (Dempster et al., 1977). It does not require to compute the gradient of the incomplete data likelihood nor the inverse of the complete data Fisher information matrix. Under some mild assumptions, it is shown in Cappé and Moulines (2009) that, even when the model is misspecified, the algorithm converges to the set of stationary points of the Kullback–Leibler divergence between the observed likelihood (which does not necessarily belongs to the statistical model) and the incomplete data likelihood. For a given value of the parameter $\theta \in \Theta$, we denote by $\pi_\theta(\cdot \mid \mathbf{Y}_n)$ the posterior distribution of the missing data $(I_n, \mathbf{X}_n)$, given the observation $\mathbf{Y}_n$. The online EM (Cappé and Moulines, 2009) is initiated with an initial guess $\hat{\theta}_0 \in \Theta$. At the nth iteration, the E-step consists in computing the function $\hat{\mathrm{Q}}_n : \Theta \to \mathbb{R}$ defined recursively for all $n > 0$ by:

$$\hat{\mathrm{Q}}_n(\theta) = \hat{\mathrm{Q}}_{n-1}(\theta) + \varrho_n \left(\mathbb{E}_{\hat{\theta}_{n-1}} \left[\log L_\theta(I_n, \mathbf{X}_n, \mathbf{Y}_n) \mid \mathbf{Y}_n \right] - \hat{\mathrm{Q}}_{n-1}(\theta) \right), \tag{12}$$

where $\mathbb{E}_{\hat{\theta}_{n-1}}(\cdot \mid \mathbf{Y}_n)$ stands for the conditional expectation under $\pi_{\hat{\theta}_{n-1}}(\cdot \mid \mathbf{Y}_n)$, $\{\varrho_n\}_{n>0}$ is a decreasing sequence of positive step sizes, with $\varrho_1 = 1$, such that $\hat{\mathrm{Q}}_0$ needs not be specified. In the M-step, the next estimate $\hat{\theta}_n$ is obtained by maximizing

$$\hat{\theta}_n = \arg\max_{\theta \in \Theta} \hat{\mathrm{Q}}_n(\theta). \tag{13}$$

Under our model specification, the complete data log-likelihood belongs to a curved exponential family. Indeed, for a given parameter $\theta \in \Theta$, $\log \mathrm{L}_\theta$ can be written as

$$\log \mathrm{L}_\theta(I, \mathbf{X}, \mathbf{Y}) = t(\theta) + \langle r(\theta), S(I, \mathbf{X}, \mathbf{Y}) \rangle, \tag{14}$$

where the function t is given by

$$t(\theta) = \log \frac{a^a}{\mathcal{G}(a)} - \frac{|\Omega|}{2} \log 2\pi\sigma^2 - d_\beta \log 2\pi,$$

$\mathcal{G}$ is the Gamma function, $S(I, \mathbf{X}, \mathbf{Y}) = (S_1(I, \mathbf{X}, \mathbf{Y}), \ldots, S_C(I, \mathbf{X}, \mathbf{Y}))$, such that for all $j \in \{1, \ldots, C\}$

$$S_j(I, \mathbf{X}, \mathbf{Y}) = \delta_{I,j} \left(1, \lambda\phi_\beta^T\mathbf{Y}, \lambda^2\phi_\beta^T\phi_\beta, \beta\beta^T, \|\mathbf{Y}\|^2, \lambda, \log\lambda\right)$$

and the functions $r(\theta) = (r_1(\theta), \ldots, r_C(\theta))$ are defined as:

$$r_j(\theta) = (1/2) \left(2\log(\omega_j) - \log\det\Gamma_j, 2\sigma^{-2}\boldsymbol{\alpha}_j, -\sigma^{-2}(\boldsymbol{\alpha}_j\boldsymbol{\alpha}_j^T), -{\Gamma_j^{-1}}^T, -\sigma^{-2}, -2a, 2(a-1)\right).$$

As a consequence, the two steps of the online EM consist in (i) computing for all $j \in \{1, \ldots, C\}$ the stochastic approximation (SA) recursion

$$\hat{s}_{n,j} = \hat{s}_{n-1,j} + \varrho_n \left(\bar{s}_{n,j}(\mathbf{Y}_n; \hat{\theta}_{n-1}) - \hat{s}_{n-1,j}\right), \tag{15}$$

where $\bar{s}_{n,j}(\mathbf{Y}_n; \hat{\theta}_{n-1}) = \mathbb{E}_{\hat{\theta}_{n-1}}\left[S_j(I_n, \mathbf{X}_n, \mathbf{Y}_n) \mid \mathbf{Y}_n\right]$ and (ii) updating the parameters according to

$$\hat{\theta}_n = \arg\max_{\theta\in\Theta} \left\{ t(\theta) + \sum_{j=1}^{C} \left\langle r_j(\theta), \hat{s}_{n,j} \right\rangle \right\}. \tag{16}$$

The maximization is in closed form. However, this algorithm remains essentially of theoretical interest, since in many situations the conditional expectation $\bar{s}_{n,j}(\mathbf{Y}_n; \hat{\theta}_{n-1})$ is not analytically tractable. This is the case in our model. Intractable E-steps have already been addressed for batch EM algorithms. In Delyon et al. (1999), the authors proved the convergence of the Stochastic Approximation EM (SAEM) algorithm in which the E-step is replaced by a stochastic approximation making use of realizations of the missing data generated according to the posterior distribution. Still, extending the SAEM algorithm to the online setup is not feasible in our case. Indeed, independent and identically distributed (*i.i.d.*) samples from $\pi_{\hat{\theta}_{n-1}}(\cdot \mid \mathbf{Y}_n)$ cannot be simulated. An alternative to the SAEM algorithm, known as MCMC-SAEM, was proposed in Kuhn and Lavielle (2004): the authors suggested to use Markov chain Monte Carlo (MCMC) methods (see Andrieu et al., 2003 for an introduction) to obtain samples from the posterior distribution.

In this paper, we adapt this approach to the sequential setting outlined above leading to the MCoEM (Monte Carlo online EM) algorithm. It is a 3-step iterative algorithm. Given the current fit of parameter $\hat{\theta}_{n-1}$ and a new observation $\mathbf{Y}_n$, the algorithm proceeds as follows:

(1) *simulation step*: simulate, using a Markov kernel K_n that admits $\pi_{\hat{\theta}_{n-1}}(\cdot \mid \mathbf{Y}_n)$ as stationary distribution, a Markov chain $\{I_n[k], \mathbf{X}_n[k]\}_{k>0}$,

(2) *stochastic approximation step*: update for each class $j \in \{1, \ldots, C\}$, the complete data sufficient statistics using the following recursion

$$\tilde{s}_{n,j} = \tilde{s}_{n-1,j} + \varrho_n \left(\frac{1}{m_n} \sum_{k=1}^{m_n} S_j(I_n[k], \mathbf{X}_n[k], \mathbf{Y}_n) - \tilde{s}_{n-1,j} \right), \tag{17}$$

where m_n is the number of MCMC iterations performed at the nth iteration of the MCoEM algorithm,

(3) *maximization step*: update the parameter $\hat{\theta}_n$ by maximizing the function:

$$\hat{\theta}_n = \arg\max_{\theta\in\Theta} \left\{ t(\theta) + \sum_{j=1}^{C} \left\langle r_j(\theta), \tilde{s}_{n,j} \right\rangle \right\}. \tag{18}$$

For numerical stability, it is recommended not to update the parameter $\hat{\theta}_n$ at each iteration, especially in the first iterations of the algorithm (see discussion in Section 5). MCoEM updates $\hat{\theta}_n$ according to a user-defined update schedule $\mathfrak{N} \subset \mathbb{N}$. Algorithm 1 provides a pseudo-code representation of MCoEM.

4. Sampling from the missing data joint posterior distribution

In this section, we construct a transition kernel K to sample the target distribution $\pi_\theta(\cdot \mid \mathbf{Y})$ (for notational simplicity, the iteration index n of the EM algorithm is omitted in this section).

Remark 1. At this stage, one might legitimately wonder why special care must be taken when choosing K, while valid MCMC routines are by now well established and available. Having a closer look at the target distribution dismisses resorting to standard MCMC methods such as the Gibbs sampler (Gelfand and Smith, 1990; Geman and Geman, 1984) to simulate samples from $\pi_\theta(\cdot \mid \mathbf{Y})$. Indeed, the target distribution is not defined on the product space $(\mathbb{I}, \mathbb{X})$ but on the following union of spaces $(\mathbb{I} = 1, \mathbb{X}) \cup \cdots \cup (\mathbb{I} = C, \mathbb{X})$. This is because, in our framework, the deformation $\mathbf{X}$ should always be consistent with the class of the observation it applies to.

Algorithm 1 Monte Carlo online EM

Input:
- Initial guess: $\hat{\theta}_0 \in \Theta$
- A stream of observations: $\mathbf{Y}_1, \mathbf{Y}_2, \ldots$
- Parameter update schedule: $\mathfrak{N} \subseteq \mathbb{N}$
- An iteration counter n, initialized to 0
- A sequence of positive step sizes $\{\varrho_1, \varrho_2, \ldots\}$ with $\varrho_1 = 1$
- MCMC length schedule $\{m_1, m_2, \ldots\}$

When a new observation $\mathbf{Y}$ is available **do**
  Increment the iteration counter: $n = n + 1$
  Simulation step: Sample m_n missing data $\{I_n[k], \mathbf{X}_n[k]\}_{k=1}^{m_n}$ from a Markov chain targeting $\pi_{\hat{\theta}_{n-1}(\cdot \mid \mathbf{Y})}$
  ▶ See Algorithm 2
  SA step: Update the sufficient statistics $\tilde{s}_{n,1}, \ldots, \tilde{s}_{n,C}$ via the stochastic approximation step
  ▶ See Eq. (17)
  If $n \in \mathfrak{N}$ **then**
    Maximization step: Update the parameter estimate to $\hat{\theta}_n$
    ▶ See Eq. (18)
  else
    Set $\hat{\theta}_n = \hat{\theta}_{n-1}$
  end if

Output: A sequence of parameters $\hat{\theta}_1, \hat{\theta}_2, \ldots$.

4.1. MCMC on an extended state space

We now explain the approach we followed. The basic idea, stemming from Carlin and Chib (1995), is to specify a joint distribution over the class index I and auxiliary variables $\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_C$, where for all $j \in \{1, \ldots, C\}$, $\tilde{\mathbf{X}}_j \in \mathbb{X}$ is a deformation parameter associated to the class $\mathcal{C}_j$. We stress that, in this approach, we sample at each iteration deformation parameters for each class. To specify the joint distribution, we introduce the *pseudo-priors* or *linking densities*, denoted $\{\kappa_{\theta,j}\}_{j=1}^{C}$. Note that whereas the knowledge of the normalizing constant is not required for an MCMC algorithm, the normalizing constant of the pseudo-priors are assumed to be known, *i.e.* the pseudo-priors $\{\kappa_{\theta,j}\}_{j=1}^{C}$ should integrate to 1. Also, it is assumed that exact sampling from the pseudo-priors is doable (and is computationally inexpensive). We define an auxiliary joint posterior density $\tilde{\pi}_\theta(\cdot \mid \mathbf{Y})$ on the product space $\mathbb{I} \times \mathbb{X} \times \cdots \times \mathbb{X}$ by:

$$\begin{aligned}\tilde{\pi}_\theta(I, \tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_C \mid \mathbf{Y}) &= \pi_\theta(I, \tilde{\mathbf{X}}_I \mid \mathbf{Y}) \prod_{j \neq I} \kappa_{\theta,j}(\tilde{\mathbf{X}}_j) \\ &\propto g_\theta(\mathbf{Y} \mid I, \tilde{\mathbf{X}}_I) p_\theta(\tilde{\mathbf{X}}_I \mid I) \omega_I \prod_{j \neq I} \kappa_{\theta,j}(\tilde{\mathbf{X}}_j), \end{aligned} \tag{19}$$

where ω_I, g_θ and p_θ are defined in (6), (10) and (11) respectively. It can be noted that the marginal of $\tilde{\pi}_\theta(\cdot \mid \mathbf{Y})$ with respect to to the auxiliary deformation parameters is the target distribution $\pi_\theta(\cdot \mid \mathbf{Y})$:

$$\pi_\theta(I, \mathbf{X} \mid \mathbf{Y}) = \int \cdots \int \tilde{\pi}_\theta(I, \tilde{\mathbf{x}}_{1:I-1}, \mathbf{X}, \tilde{\mathbf{x}}_{I+1:C} \mid \mathbf{Y}) \mathrm{d}\tilde{\mathbf{x}}_{-I}, \tag{20}$$

where for all $(i, j) \in \mathbb{I}^2$, such that $i < j$, $a_{i:j} = (a_i, a_{i+1}, \ldots, a_j)$ and for all $i \in \mathbb{I}$, $a_{-i} = \{a_j\}_{j=1, j \neq i}^{C}$. Remarkably, this property does not depend on the choice of pseudo-priors.

A Metropolis-within-Gibbs sampler targeting $\tilde{\pi}_\theta(\cdot \mid \mathbf{Y})$ is used to simulate a Markov chain $(I[k], \tilde{\mathbf{X}}_1[k], \ldots, \tilde{\mathbf{X}}_C[k])$ on the product space $(\mathbb{I} \times \mathbb{X} \times \cdots \times \mathbb{X})$. Suppose the Markov chain is at state $(I, \tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_C)$, the so-called full conditional posterior distributions required for the Gibbs sampler are:

$$\tilde{\pi}_\theta(I \mid \tilde{\mathbf{X}}_{1:C}, \mathbf{Y}) \propto g_\theta(\mathbf{Y} \mid I, \tilde{\mathbf{X}}_I) p_\theta(\tilde{\mathbf{X}}_I \mid I) \omega_I \prod_{j \neq I} \kappa_{\theta,j}(\tilde{\mathbf{X}}_j), \tag{21}$$

$$\tilde{\pi}_\theta(\tilde{\mathbf{X}}_j \mid I, \tilde{\mathbf{X}}_{-j}, \mathbf{Y}) \propto \begin{cases} g_\theta(\mathbf{Y} \mid I, \tilde{\mathbf{X}}_I) p_\theta(\tilde{\mathbf{X}}_I \mid I) = \pi_\theta(\tilde{\mathbf{X}}_I \mid I, \mathbf{Y}), & j = I, \\ \kappa_{\theta,j}(\tilde{\mathbf{X}}_j), & j \neq I. \end{cases} \tag{22}$$

From (21) and (22), it can be seen that sampling the class index and the auxiliary deformations from their respective full conditional posterior distribution is straightforward. However, since sampling the new parameter from the current class cannot be achieved directly, a Random Walk Metropolis–Hastings (RWMH) (Metropolis et al., 1953) kernel $P_\theta(\tilde{\mathbf{X}}_I; \cdot \mid$

$\tilde{\mathbf{X}}_{-I}, I, \mathbf{Y})$ having $\pi_\theta(\cdot \mid I, \mathbf{Y})$ as its stationary distribution is applied r times to $\tilde{\mathbf{X}}_I$ to generate $\tilde{\mathbf{X}}'_I$. The Markov chain transition writes:

(i) $I' \sim \tilde{\pi}_\theta(\cdot \mid \tilde{\mathbf{X}}_{1:C}, \mathbf{Y})$,
(ii) $\tilde{\mathbf{X}}'_j \sim \kappa_{\theta,j}$, for $j \neq I'$,
(iii) $\tilde{\mathbf{X}}'_{I'} \sim P^r_\theta(\tilde{\mathbf{X}}_{I'};\ \cdot \mid \tilde{\mathbf{X}}'_{-I'}, I', \mathbf{Y})$,

and the transition kernel $\tilde{K}^{\mathrm{CC}}$ may thus be expressed as:

$$\tilde{K}^{\mathrm{CC}}(I', \mathrm{d}\tilde{\mathbf{X}}'_{1:C} \mid I, \tilde{\mathbf{X}}_{1:C}) = \tilde{\pi}_\theta(I' \mid \tilde{\mathbf{X}}_{1:C}, \mathbf{Y}) P^r_\theta(\tilde{\mathbf{X}}_{I'}; \mathrm{d}\tilde{\mathbf{X}}'_{I'} \mid \tilde{\mathbf{X}}'_{-I'}, I', \mathbf{Y}) \prod_{j \neq I} \kappa_{\theta,j}(\mathrm{d}\tilde{\mathbf{X}}'_j). \tag{23}$$

In stationary regime, the Markov chain $\{I[k], \tilde{\mathbf{X}}_1[k], \ldots, \tilde{\mathbf{X}}_C[k]\}_{k>0}$, simulated through a Metropolis-within-Gibbs algorithm, provides samples from $\tilde{\pi}_\theta(\cdot \mid \mathbf{Y})$. However, only the marginal samples $\{I[k], \mathbf{X}[k] = \tilde{\mathbf{X}}_{I[k]}\}_{k>0}$, distributed under $\pi_\theta(\cdot \mid \mathbf{Y})$ (20), are of interest and will be used in the approximation of the E-step of the MCoEM algorithm (17). Pseudo-code of the Markov chain simulation algorithm is reported in Algorithm 2.

Algorithm 2 Markov chain simulating missing data

Input:
- An observation: $\mathbf{Y}$
- A parameter estimate: θ
- Number of components: C
- Length of the Markov chain: m
- Number of RWMH iterations: r

Specification of the pseudo-prior densities $\kappa_{\theta,1}, \ldots, \kappa_{\theta,2}$
  ▶ See Section 4.2
Set $\tilde{\mathbf{X}}_j[0] \sim \kappa_{\theta,j}$ for $j = 1, \ldots, C$
for $k = 1, \ldots, m$ **do**
  Class sampling: $I[k] \sim \tilde{\pi}_\theta(I \mid \tilde{\mathbf{X}}_{1:C}[k-1], \mathbf{Y})$
    ▶ See Eq. (21)
  Let $i = I[k]$
  Random Walk Metropolis–Hastings move:

$$\tilde{\mathbf{X}}_i[k] \sim P^r_\theta(\tilde{\mathbf{X}}_i[k-1];\ \cdot \mid i, \mathbf{Y})$$

  for $j \in \{1, \ldots, C\} \backslash \{i\}$ **do**
    Pseudo-prior update: $\tilde{\mathbf{X}}_j[k] \sim \kappa_{\theta,j}$
  end for
  Set $\mathbf{X}[k] = \tilde{\mathbf{X}}_i[k]$
end for
Output: A Markov chain $(I[1], \mathbf{X}[1], \ldots, I[m], \mathbf{X}[m])$.

4.2. Choice of the pseudo-prior densities

The specification of the linking densities is essential for sampling efficiency. Ideally, these densities should be close to the marginal posterior: for all $j \in \{1, \ldots, C\}$, the density $\mathbf{X} \to \kappa_{\theta,j}(\mathbf{X})$ should be chosen as a proxy to $\mathbf{X} \to \pi_\theta(\mathbf{X} \mid j, \mathbf{Y})$. An idea is for instance to set the pseudo-prior density as a Gaussian approximation of the target density. Such an approximation can be obtained using the Laplace method (Wolfinger, 1993) or other approximate Bayesian sampling method. Under the (weak) assumption that the function $\mathbf{X} \to \pi_\theta(\mathbf{X} \mid j, \mathbf{Y})$ admits a maximum,

$$\mathbf{X}^\star_j = \arg\max_{\mathbf{X} \in \mathbb{X}} \pi_\theta(\mathbf{X} \mid j, \mathbf{Y}), \tag{24}$$

the Taylor-expansion of $\log \pi_\theta(\cdot \mid j, \mathbf{Y})$ at $\mathbf{X}^\star_j$ writes:

$$\log \pi_\theta(\mathbf{X} \mid j, \mathbf{Y}) = \log \pi_\theta(\mathbf{X}^\star_j \mid j, \mathbf{Y}) + \frac{1}{2}(\mathbf{X} - \mathbf{X}^\star_j)^T H_j (\mathbf{X} - \mathbf{X}^\star_j) + o(\|\mathbf{X} - \mathbf{X}^\star_j\|^2), \tag{25}$$

where for all $j \in \{1, \ldots, C\}$, H_j is the Hessian matrix, whose coefficients are given for all $(q, r) \in \{1, \ldots, d_{\mathbf{X}}\}^2$ by:

$$[H_j]_{q,r} = \left. \frac{\partial^2}{\partial \mathbf{X}_q \partial \mathbf{X}_r} \log \pi_\theta(\mathbf{X} \mid j, \mathbf{Y}) \right|_{\mathbf{X} = \mathbf{X}^\star_j}. \tag{26}$$

Note that for better readability, for all $j \in \{1, \ldots, C\}$, the dependence of the linking densities $\kappa_{\theta,j}$, on the parameters $\mathbf{X}_j^\star$, H_j on $\mathbf{Y}$ and θ is not made explicit in these notations, but does exist.

The previous discussion suggests that $\mathcal{N}_{d_\mathbf{X}}(\mathbf{X}_j^\star, -H_j^{-1})$ is a sensible candidate for $\kappa_{\theta,j}$. The pseudo-priors parameters $\mathbf{X}_j^\star$ may be obtained using standard nonlinear optimization methods. Since $\mathbf{X}^\star$ is only used in the pseudo-prior specification, the precision of the optimizer does not matter much and simple heuristics can be used (see related discussion in Section 5).

Remark 2. Our proposed kernel shares some similarities with that proposed in Allassonnière and Kuhn (2010), which also makes use of auxiliary variable $\{\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_C\}$. These authors propose to first sample the class index I from $\pi_\theta(\cdot \mid \mathbf{Y})$ and then draw $\mathbf{X} \sim \pi_\theta(\cdot \mid I, \mathbf{Y})$. However, since sampling the class index from the posterior distribution is not doable (indeed $\pi(I = j \mid \mathbf{Y}) \propto \pi_\theta(j, \mathbf{Y})$ which is not analytically tractable), auxiliary variables $\{\tilde{\mathbf{X}}_1[k], \ldots, \tilde{\mathbf{X}}_C[k]\}_{k>0}$ are sampled from C independent Markov chains each targeting $\pi_\theta(\cdot \mid j, \mathbf{Y})$, $j \in \{1, \ldots, C\}$, in an attempt to approximate the posterior weights $\{\pi_\theta(j \mid \mathbf{Y})\}_{j=1}^C$. These approximate weights allow to sample I and then parameter samples $\{\mathbf{X}[k]\}_{k>0}$ are drawn using a Markov chain targeting $\pi_\theta(\cdot \mid I, \mathbf{Y})$. Since the inference in Allassonnière and Kuhn (2010) is conducted in a batch learning setup, this scheme is computationally intensive. Indeed, at each iterations, C Markov chains $\{\tilde{X}_1[k], \ldots, \tilde{X}_C[k]\}_{k>0}$ per observation need to be sampled.

5. Numerical illustration

We evaluate the performance of our online learning algorithm by inferring two types of data: growth velocity curves and handwritten digits. These two examples illustrate the flexibility, the stability and the computational effectiveness of the proposed MCoEM. MCoEM is then compared to an equivalent SAEM algorithm on the handwritten digits templates extraction task.

5.1. Growth velocity curve study

The growth velocity curve example is a classical benchmark in curve registration; see Ramsay (2006), Zhong (2008), Dimeglio et al. (2014) and Wu and Hitchcock (2016). It is used here for illustrative purposes, because the rationale of the model is easy to grasp. The growth curves are obtained from the Berkeley Growth Study data (Tuddenham and Snyder, 1954) and display the evolution of the growth velocity between 2 and 18 years, for 39 boys and 54 girls; see Fig. 1. Even though each observation is known to arise from either a boy or a girl, this information is unused, as MCoEM is designed to perform unsupervised inference on mixture models. The objective of the algorithm is therefore to retrieve a standard growth profile for boys and girls from the unlabeled set of growth velocity curves. The growth velocity curves, plot the growth velocity of individuals observed at $|\Omega| = 31$ landmarks $\Omega = \{u_1, \ldots, u_{|\Omega|}\}$, irregularly spaced, such that for all $s \in \{1, \ldots, |\Omega|\}$, $2 \leq u_s \leq 18$.

5.1.1. Deformable template model

Growth profiles may vary from an individual to another, both as a function of the time and in amplitude. The algorithm aims to extract templates for the growth velocity curves: it associates to each observation $\mathbf{Y}_n$ a monotonically increasing time warping function $u \mapsto D(u, \beta_n)$ as well as a global scaling parameter λ_n. We consider a mixture model with $C = 2$, implying that we aim at retrieving templates for boys and girls growth velocity separately: the class index $I_n \in \{1, 2\}$ models the boys and girls clusters. In this illustration, the template is a function $f_{\boldsymbol{\alpha}_i}$ ($i \in \{1, 2\}$) defined on an open segment $\mathbb{U} = (u_i, u_f) = (2, 18)$ parameterized as:

$$f_{\boldsymbol{\alpha}_i}(u) = \sum_{\ell=1}^{m} \alpha_{i,\ell} \phi_\ell(u), \quad (\alpha_{i,1}, \ldots, \alpha_{i,m}) \in \mathcal{A} = \mathbb{R}^{+m}, \tag{27}$$

where $\{\phi_\ell\}_{\ell=1}^m$ is set as $u \mapsto \phi_\ell(u) = \exp(\nu_\ell{}^{-2}(u - r_\ell)^2)$, where $\{r_\ell\}_{\ell=1}^m$ are regularly spaced landmark points in $\mathbb{U}$. The choice of $\{\phi_\ell\}_{\ell=1}^m$ and $\mathcal{A}$ ensures that the template function $u \mapsto f_{\boldsymbol{\alpha}_i}(u)$ is a positive function, which is a natural constraint for growth velocity curves. For all $\ell \in \{1, \ldots, m\}$, the bandwidth of ϕ_ℓ is set as $\nu_\ell{}^2 = -\min_{u \in \Omega \setminus \{r_\ell\}} \|r_\ell - u\|^2 / \log \varepsilon$, where $\varepsilon \in (0, 1)$ is the value of ϕ_ℓ at the nearest design point of r_ℓ. This choice of bandwidth enables to take into account the irregularly spaced measurement points in Ω. In this implementation, we used $m = 35$, so that kernels $\phi_1, \phi_2, \ldots$ are centered on landmarks distant from a 6-month interval and $\varepsilon = 0.1$. The deformable template model (1) simply writes for all $u \in \mathbb{U}$:

$$Y_n \in \mathcal{C}_i, \quad Y_n(u) = \lambda_n f_{\boldsymbol{\alpha}\mathbf{i}} \circ D(u, \beta_n) + \sigma W_n(u).$$

In this setting, the time warping function $u \mapsto D(u, \beta)$ is monotonically increasing and should satisfy $D(u_i, \beta) \geq u_i$ and $D(u_f, \beta) \leq u_f$ (indeed, outside (u_i, u_f), the template vanishes (27)). In order to satisfy these constraints, we write $D(\cdot, \beta)$ as:

$$D(u, \beta) = u_i + (u_f - u_i) H(u, \beta), \tag{28}$$

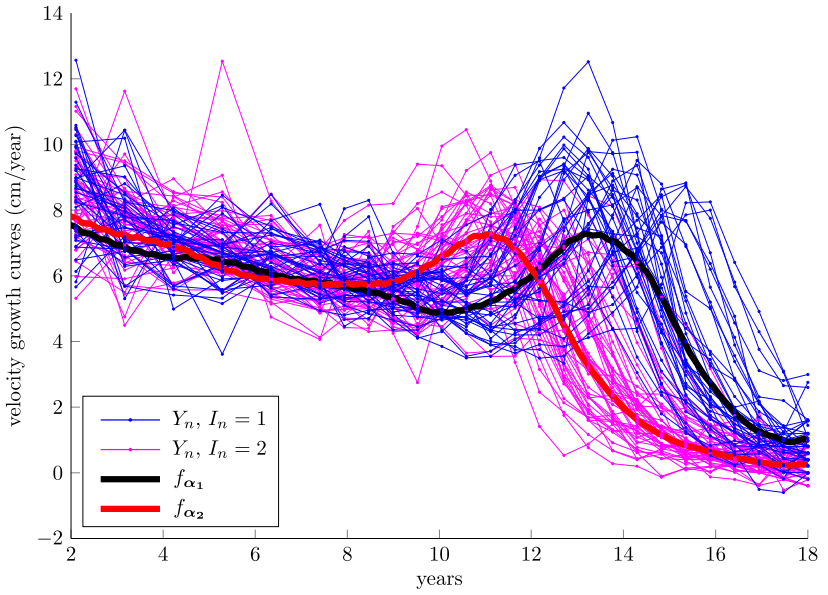


Fig. 1. Growth velocity samples and templates extraction obtained through 1000 iterations of the MCoEM algorithm.

where $H(\cdot, \beta)$ is modeled as proposed in Ramsay and Li (1998) with:

$$H(u, \beta) = \frac{\int_{u_i'}^{u} \exp\left[\sum_{k=1}^{d_\beta} \beta_k \psi_k(v)\right] \mathrm{d}v}{\int_{u_i'}^{u_f'} \exp\left[\sum_{k=1}^{d_\beta} \beta_k \psi_k(v)\right] \mathrm{d}v}, \tag{29}$$

where $u_i' \leq u_i$ and $u_f' \geq u_f$ allow to satisfy the constraints stated above. For all k in $\{1, \ldots, d_\beta\}$, $\beta_k \in \mathbb{R}$ and $\{\psi_k\}_{k=1}^{d_\beta}$ is a dictionary of Gaussian kernels centered on the landmark points $\{q_k\}_{k=1}^{d_\beta}$ with the same bandwidth τ^2. In this implementation, we set $u_i' = 0, u_f' = 20$ and use $d_\beta = 20$ regularly spaced landmark points such that $q_1 = u_i'$ and $q_{d_\beta} = u_f'$; the kernel variance is set to $\tau^2 = 1$. Moreover, the prior distribution (10) of β is set with a mean equals to $(1, \ldots, 1)^T$ and for all $j \in \{1, 2\}$ a covariance matrix Γ_j parameterized by the variance γ_j, such that $\Gamma_j = \gamma_j^2 \mathrm{Id}_{d_\beta}$. The estimate $\hat{\gamma}_{j,n}^2$ of γ_j^2 after $n = 1000$ iterations is $\hat{\gamma}_{1,1000}^2 = 0.08$ and $\hat{\gamma}_{2,1000}^2 = 0.07$. The hyper-parameter of the prior distribution for the scaling parameter λ is set to $a = 10$.

5.1.2. Sampling the missing data

Figs. 2–4 illustrate the sampling scheme proposed in Section 4, taking place at a given iteration n of the MCoEM algorithm (the index n is omitted hereafter). For $j \in \{1, 2\}$, the auxiliary variable $\tilde{\mathbf{X}}_j$ consists in $\tilde{\mathbf{X}}_j = (\lambda_j, \beta_j)$. In Fig. 2, green dots represent an observation Y along with the templates in plain curves (boys on the top panel and girls on the bottom panel). In each panel, the dashed curves illustrate different realizations of the distorted template under the action of deformation parameters $\tilde{\mathbf{X}}_1[k] = (\beta_1[k], \lambda_1[k])$ and $\tilde{\mathbf{X}}_2[k] = (\beta_2[k], \lambda_2[k])$ sampled using the kernel $\tilde{K}^{\mathrm{CC}}$. For each new observation $\mathbf{Y}$, we used 300 iterations of the Markov chain detailed in Section 4.1, discarding the first 100 states for burn-in. The pseudo-priors κ_1 and κ_2 were set as Gaussian distributions, as specified in Section 4.2. For $j \in \{1, 2\}$, the mean $(\lambda_j^\star, \beta_j^\star)$ were obtained through a quasi-Newton optimization method (with an early stopping rule, because the precision of the fit does not matter much). For computational efficiency, the pseudo-prior κ_j covariance matrix was set as $\hat{\Gamma}_{j,n} = \hat{\gamma}_{j,n}^2 \mathrm{Id}_{d_\beta}$ (which is the jth class prior covariance matrix estimate). Even though, the pseudo-prior distributions provide inappropriate deformation parameters (see some samples from $D(\cdot, \beta_1[k])$ on the top panel of Fig. 2), they nevertheless achieve their two-fold target, namely (i) allowing to switch between models as illustrated in Fig. 3 and (ii) sampling deformations that are consistent with $\mathbf{Y}$: the distorted templates tend to match the observation. Fig. 3 shows two warping functions $D(\cdot, \beta_1[k])$ and $D(\cdot, \beta_2[k])$ corresponding to the samples $\beta_1[k]$ and $\beta_2[k]$ obtained at the $k = 300$th iteration of the Markov chain. This shows that, in order to register the template with the observation, the boys time warping function (in black, parameterized by β_1) accelerates the time from 9 years old onwards much faster than its girls counterpart (in red, parameterized by β_2). This is an evidence that this observation is more likely to arise from a girl record. The sampling of the cluster index (21) makes use of the complete data log-likelihood and promotes models involving small deformations. Therefore, the class $I = 2$ is more likely as confirmed by Fig. 4 representing the class sampling scheme throughout the 300 MCMC iterations.

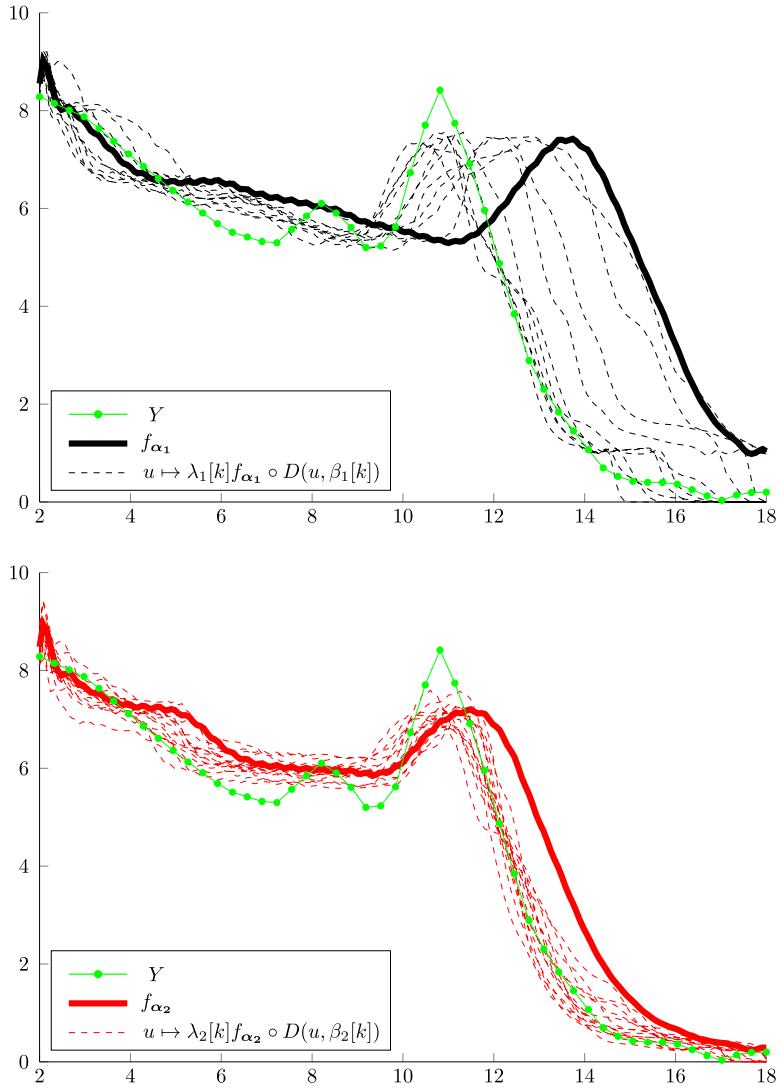


Fig. 2. Sampling of the hidden data posterior distribution. $\{\lambda_1[k], \beta_1[k], \lambda_2[k], \beta_2[k], I[k]\}$ for some $k \in \{101 \ldots, 300\}$ are samples from the Markov chain produced by $\tilde{K}^{\text{CC}}$ that admits $\tilde{\pi}_\theta(\cdot \mid \mathbf{Y})$ (19) as stationary distribution. The sampled deformation/scale is then applied to the template of the class it corresponds to (the thick black/red line), yielding a distorted template (the dashed black/red line) that tends to match the observation (the green dotted line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.1.3. Template estimation

Starting with two $m = 35$ dimensional random vectors $\hat{\boldsymbol{\alpha}}_{1,0}$ and $\hat{\boldsymbol{\alpha}}_{2,0}$, the two templates $f_{\hat{\boldsymbol{\alpha}}_{1,1000}}$ and $f_{\hat{\boldsymbol{\alpha}}_{2,1000}}$, displayed in Fig. 1 were obtained after $N = 1000$ iterations of the MCoEM algorithm. Since a limited number of observations are available, each observation is processed several times, drawn at random throughout the iterations. The templates show that the girls reach the pubertal growth spurt earlier (between 11 and 12 years) than boys (between 13 and 14 years). Moreover, we notice that the boys growth velocity profile features a pre-pubertal dip more pronounced than for the girls. The estimated templates are comparable to those obtained for example in Wu and Hitchcock (2016) (see Figure 8) using a similar model in a fully Bayesian inference framework.

5.2. Handwritten digits template extraction

We apply MCoEM to a collection of handwritten digits, the US postal database. It contains $N = 1000$ samples of each handwritten digit from 0 to 9 and each observation is a 16×16 pixel image. The USPS digits data were gathered at the Center of Excellence in Document Analysis and Recognition (CEDAR) at SUNY Buffalo, as part of a project sponsored by the US Postal Service; see Hull (1994). The main difficulty with these data stems from the geometric dispersion *within each class* of digit. Two sources of variability are considered:

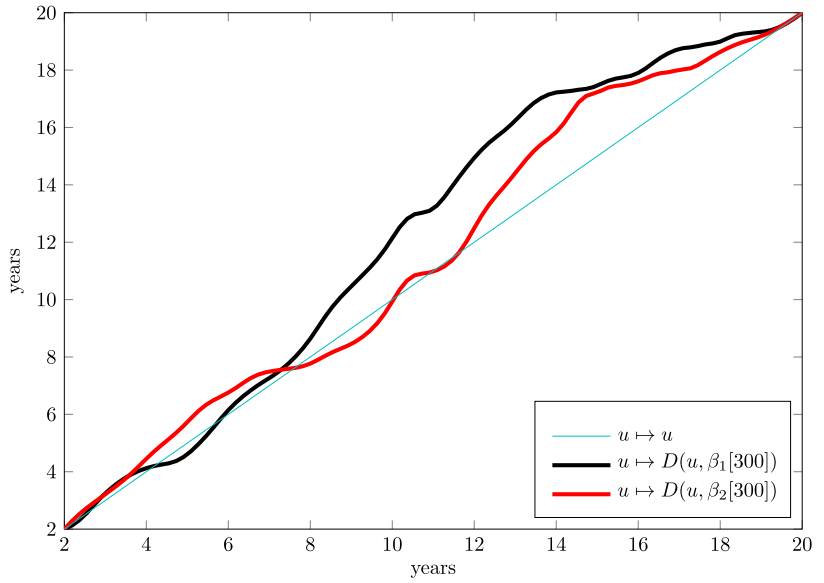


Fig. 3. Time warping functions for the deformation parameters β_1 and β_2 sampled at the last iteration ($k = 300$) of the Markov chain produced by $\tilde{K}^{\text{CC}}$ that admits $\tilde{\pi}_\theta(\cdot \mid \mathbf{Y})$ (19) as stationary distribution; see Fig. 2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

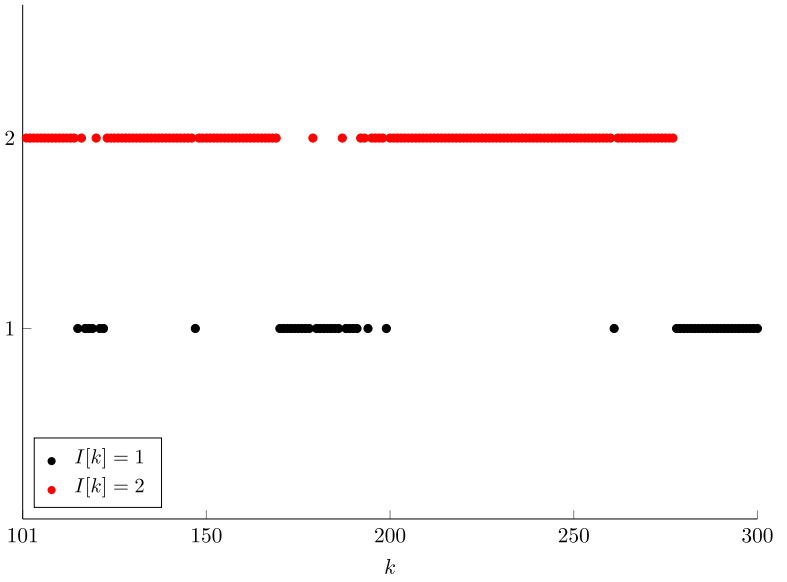


Fig. 4. Class sampling $\{I[1], \ldots, I[300]\}$ from the Markov chain produced by $\tilde{K}^{\text{CC}}$ that admits $\tilde{\pi}_\theta(\cdot \mid \mathbf{Y})$ (19) as stationary distribution. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(i) The first type is assumed meaningful, since intrinsically related to the class of digit, and MCoEM seeks to learn them: the templates. A digit may indeed need more than a single prototype shape to be efficiently modeled by a mixture of deformable templates. For example, a digit two may be written with or without a loop in the lower left-hand corner and a digit seven may feature an horizontal bar on the diagonal line.

(ii) The second type is regarded as nuisances resulting from the presentation context and are deemed irrelevant to identify the class of an observation. They consist of small local deformations and global deformations such as a rotations, homotheties and translations. Such nuisances result from the size of the pen used, different handwriting skills, digits being partially censored by the observation window, etc.

5.2.1. Deformable template model

An observation $\mathbf{Y}_n$ is a 16×16 matrix, regarded as a $|\Omega| = 256$ dimensional vector, whose coordinates correspond to the photometry of a fixed set of pixels, $(u_1, \ldots, u_{|\Omega|})$, such that for all $s \in \{1, \ldots, |\Omega|\}$, $u_s \in (-1, 1) \times (-1, 1)$. The raw

database consists of noise-free observations, such that for all $s \in \{1, \dots, |\Omega|\}$, $\mathbf{Y}_{n,s} \in (0, 1)$. To make the problem more challenging, an additive Gaussian noise $W_s = \sigma\epsilon$, where $\sigma = 0.2$ and $\epsilon \sim \mathcal{N}(0, 1)$, is added to each pixel $\mathbf{Y}_{n,s}$ (see Fig. 6(a)).

A template f is a function defined on $\mathbb{U} = \mathbb{R}^2$. The dictionary of functions $\{\phi_\ell\}_{\ell=1}^m$ is set as Gaussian kernels with $m = 256$. The landmark points $\{r_\ell\}_{\ell=1}^m$ are regularly spaced in the square $(-1, 1) \times (-1, 1)$ and the kernel ϕ_ℓ is defined as $u \mapsto \phi_\ell(u) = \exp(\nu^{-2}(u - r_\ell)^2)$ with $\nu = 0.2$.

Contrary to the growth velocity curve case, where growth profiles feature different scales, since the images are scaled in $(0, 1)$, the scale dispersion in the measurement space is limited. As a consequence, using a scaling factor λ_n is not relevant. To each image $\mathbf{Y}_n$ is associated a class $I_n \in \{1, \dots, C\}$ and a deformation parameter β_n, such that a template can be geometrically deformed under the action of a function $u \mapsto D(u, \beta_n)$. We consider two complementary types of deformation:

- A rigid deformation $u \mapsto T(u, \upsilon_n)$ where υ_n parameterizes rotations, homotheties and translations. Indeed, the templates need to be allowed to rotate and to be translated in space, in order to match the observations and in particular those which are partially censored by the observation window. Homotheties allow to zoom in or to zoom out the templates. In this case, υ_n is a 6-dimensional real vector, $\upsilon_n = (\varphi_n, \varrho_n, c_n, t_n)$, where c_n is the center of the rotation of angle φ_n and of the homotheties having ϱ_n as ratio and t_n is the translation vector. $T(\cdot, \upsilon_n)$ writes for all $u \in \mathbb{U}$:

  $$T(u, \upsilon_n) = \mathcal{R}_{\varphi_n}(\varrho_n u + t_n - c_n) + c_n,$$

  where $\mathcal{R}_{\varphi_n}$ is the rotation matrix with angle φ_n. A Gaussian prior is set on υ_n, with zero mean for the components (φ_n, c_n, t_n) and a mean one for ϱ_n. The covariance matrix is diagonal with variances set to 0.1.
- A smooth small deformation field is used to register locally a template with the observation. It is parameterized by a d_V-dimensional vector $\delta_n = (\delta_{n,1}, \dots, \delta_{n,d_V})$ and writes for all $u \in \mathbb{U}$ as

  $$V(u, \delta_n) = \sum_{k=1}^{d_V} \delta_{n,k}\psi_k(u),$$

  where for all $k \in \{1, \dots, d_V\}$, $\delta_{n,k} \in \mathbb{R}^2$ allows small displacements in the two directions. The smoothness of the deformation is enforced by the choice of functions $\{\psi_k\}_{k=1}^{d_V}$ which belongs to a dictionary of Gaussian kernels defined on $\mathbb{R}^2$ and centered on the landmark points $\{q_k\}_{k=1}^{d_V}$ with identical variance σ_L^2, such that for all $k \in \{1, \dots, d_V\}$, $\psi_k(u) = \exp\left(\sigma_V^{-2}\|u - q_k\|^2\right)$. In this implementation, we used $d_V = 36$ landmark points at the vertices of a regular grid on the square $(-0.5, 0.5) \times (-0.5, 0.5)$ and a bandwidth $\sigma_V^2 = 0.16$. As a consequence the local deformation parameter δ_n is a 72-dimensional vector. Similarly to υ_n, conditionally on $I_n = j$, a Gaussian distribution with zero mean and covariance matrix $\bar{\Gamma}_j$ is assumed for the parameter δ_n. In this implementation, for all $j \in \{1, \dots, C\}$, $\bar{\Gamma}_j$ writes $\bar{\Gamma}_j = \gamma_j^2 M$ where M is a $2K_g \times 2K_g$ fixed matrix with ones on the diagonal, 0.2 on the lower and upper diagonals and 0 everywhere else.

Hence, the parameter β_n is a 78-dimensional vector which writes $\beta_n = (\upsilon_n, \delta_n)$ and belongs to the space $\mathbb{B} = [0, 2\pi] \times \mathbb{R}^+ \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^{72}$. Finally, the deformation model writes in this setting for all $u \in \mathbb{U}$ and a parameter $\beta_n \in \mathbb{B}$ as:

$$D(u, \beta_n) = \mathcal{R}_{\varphi_n}(\varrho_n u + t_n - c_n) + c_n + \sum_{k=1}^{d_V} \delta_{n,k}\psi_k(u). \tag{30}$$

It is illustrated with Fig. 5.

5.2.2. Parameter estimation

We consider two learning setups:

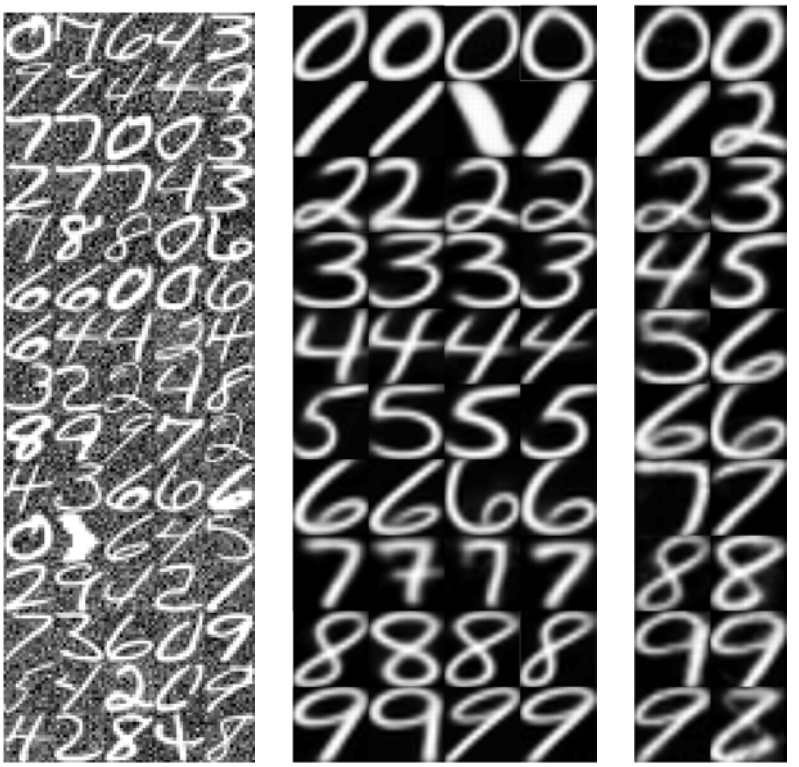
1. **Partially-supervised**: the templates are learnt for each digit separately with $C_1 = 4$ classes through $N_1 = 1000$ iterations of the MCoEM. Thus, 10 independent models are learnt and the resulting templates are reported in Fig. 6(b). We refer to this approach as partially-supervised since MCoEM deals with images of the same digit (labeled) but assigns each observation to one of the four classes describing this type of digit in an unsupervised fashion.
2. **Fully-unsupervised**: the templates are learnt from the dataset containing all the 10 digits (unlabeled), with $C_2 = 20$ classes and $N_2 = 5000$. Thus, only one model is learnt and the resulting templates are illustrated with Fig. 6(c).

The templates obtained in the two settings are similar, even though in Fig. 6(c), the algorithm makes use of a class for digits that can hardly be classified in one of the existing mixture component (template in the bottom right corner). In addition, in the fully-unsupervised scheme, the number of classes describing a digit is ruled by the learning algorithm and may not be optimal: for instance a digit two could be described with more than two clusters, whereas three classes for a digit nine are a bit excessive. For a qualitative comparison, the template shapes in Fig. 6(c) can be compared to those estimated in Nguyen et al. (2016) (Figure 3) modeled with a mixture of spatial spline regression (MSSR) and fitted with a batch EM algorithm on a similar dataset.

Following the guidelines provided in Cappé and Moulines (2009), the sufficient statistics (and consequently the parameters) should be updated for the first time once several observations have been gathered. Indeed, the sufficient statistics estimator needs to satisfy a number of constraints. In particular $\{\tilde{s}_{n,j,1}\}_{j=1}^C$ should be nonzero scalars and $\{\tilde{s}_{n,j,3}\}_{j=1}^C$ should

template $f_{\boldsymbol{\alpha}}$ | rotation of the template: $f_{\boldsymbol{\alpha}} \circ T(\cdot, \upsilon_n)$ | rotation and zoom out of the template: $f_{\boldsymbol{\alpha}} \circ T(\cdot, \upsilon_n)$ | rotation, zoom out and translation of the template: $f_{\boldsymbol{\alpha}} \circ T(\cdot, \upsilon_n)$ | rotation, zoom out, translation and small deformations of the template: $f_{\boldsymbol{\alpha}} \circ D(\cdot, \beta_n)$

Fig. 5. Distortion of a template $f_{\boldsymbol{\alpha}}$ under the action of global and local deformations.

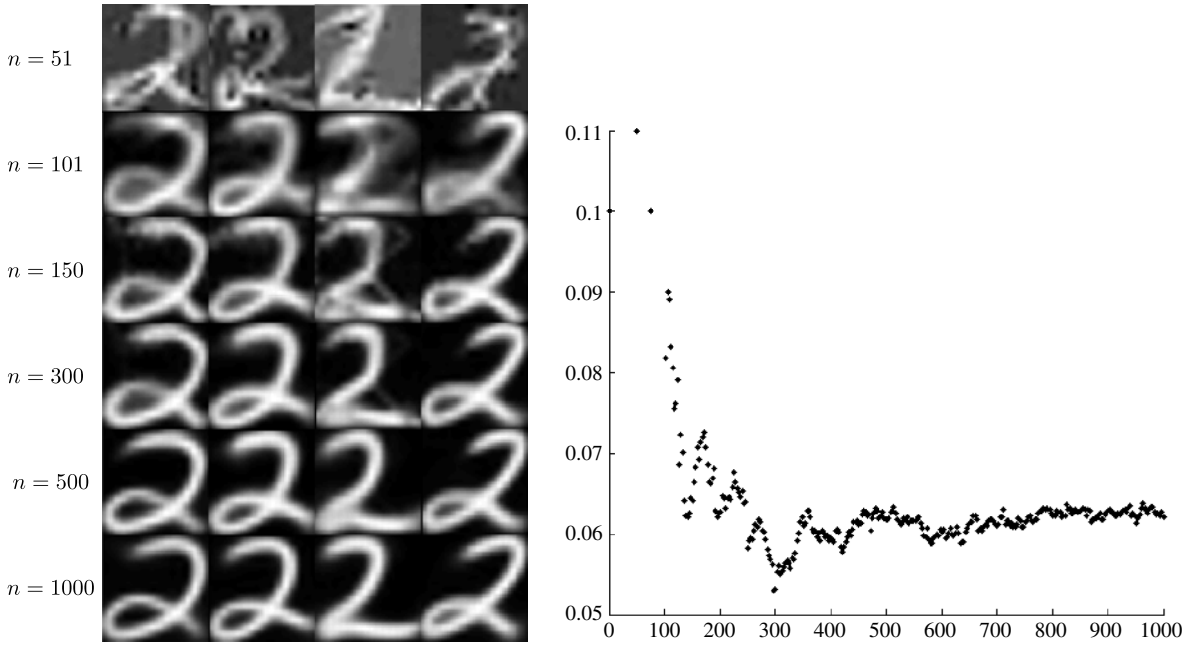


(a) Samples of handwritten digits. (b) Partially supervised scheme. (c) Fully unsupervised scheme.

Fig. 6. Templates estimated in the two different schemes: (b) partially-supervised, after $N_1 = 1000$ MCoEM iterations with $C_1 = 4$ components for each model and (c) fully-unsupervised, after $N_2 = 5000$ MCoEM iterations with $C_2 = 20$ components. In both setups, MCoEM was applied to handwritten digit images similar to those displayed in (a).

be invertible matrices. In practice, these assumptions hold, when the first update happens after $n = 50$ MCoEM iterations, the second after $n = 75$ and as soon as a new observation is available from $n = 100$ onwards. Initialization of the template parameters can potentially lead to degeneracy if one or more classes are initialized with pathologic parameters. This issue was not encountered in the partially-supervised setup probably because the class sampling is easier, the data being all observations of the same digit. The initial template parameters were thus set randomly. In the fully-unsupervised scheme however, the template parameters were set as the clusters centroid returned by a k-means clustering algorithm (using the Matlab built-in routine) applied to 50 images of the dataset drawn at random. More precisely, for all $j \in \{1, \ldots, C\}$, $\hat{\boldsymbol{\alpha}}_{0,j} = (\Phi_{O_{d_\beta}}^T \Phi_{O_{d_\beta}})^{-1} \Phi_{O_{d_\beta}}^T \mathbf{c}_j$, where Φ_β is defined in Eq. (3) and $\mathbf{c}_j$ is k-means cluster j centroid.

Fig. 7 shows the parameters estimate $\{\hat{\theta}_n\}_{n=1}^{1000}$ throughout the MCoEM algorithm for the digit two learnt separately with $C = 4$ classes (partially-supervised). The functions $\{f_{\boldsymbol{\alpha}_{n,j}}\}_{1 \leq j \leq C}$ tend progressively to usual reference shapes and each new observation available enhances the templates estimate (Fig. 7(a)).



(a) Evolution of the templates $\{f_{\hat{\alpha}_j}\}_{1\leq j\leq C}$.

(b) Evolution of the noise variance estimate $\hat{\sigma}_n^2$.

Fig. 7. Templates extraction and inference.

5.2.3. Sampling the missing data

The hidden data $\beta_n = (\upsilon_n, \delta_n)$ and I_n are simulated with $t_n = 200$ iterations if $n \leq 100$ and $t_n = 500$ iterations otherwise of the sampling scheme proposed in Section 4. This choice of m_n is motivated by the fact that when the templates are not well resolved (which occurs in the early estimates of MCoEM), a rough approximation of the conditional expectation is sufficient. Moreover, a burn-in period of 100 iterations was applied. Finally, given the high dimension of β_n, the quasi-Newton optimization methods to estimate $\{\beta_j^\star\}_{j=1}^C$ in Eq. (24) is time-consuming. Therefore, for all $j \in \{1, \ldots, C\}$ the pseudo-prior $\kappa_{\theta,j}$ parameters are set as the sample mean and covariance matrix derived from 100 iterations of a random walk targeting the posterior distribution $\pi_\theta(\cdot \mid \mathbf{Y}, I = j)$ and taking place before the first MCMC iteration.

Fig. 8(a) shows a realization of the $k = 450$th iteration of the Markov chain $\tilde{K}_n^{\mathrm{CC}}$ occurring at the $n = 600$th iteration of the MCoEM algorithm (the index n is omitted hereafter). In this scenario, we aim at extracting $C = 4$ templates of the digit 7 in a partially-supervised setting (see Fig. 6(b)). Given $I[k-1] = 3$, the auxiliary variables $\{\beta_j[k]\}_{j\neq 3}$ are sampled from the linking densities $\{\kappa_{\hat{\theta}_{n-1},j}\}_{j\neq 3}$, while $\beta_3[k]$ is simulated with $r = 20$ iterations of a Gaussian increment Random Walk Metropolis–Hastings algorithm, whose variance is adjusted to obtain an overall acceptance rate of 40% (see Andrieu et al., 2003). Iterating the Metropolis–Hastings kernel r times speeds up the convergence of the chain without changing the stationary distribution. Despite the rough approximation on the pseudo-priors parameters, Fig. 8(a) shows that the simulated deformations $\beta_j[k]$ are consistent with the observation $\mathbf{Y}_n$ for each model $j \in \{1, \ldots, C\}$. As a consequence, the Markov chain $\{I[k], \beta_1[k], \ldots, \beta_C[k]\}_{k>0}$ mixes well; see Fig. 8(b) which displays the class index samples $\{I[k]\}_{k>0}$ throughout the $t_n = 500$ MCMC iterations. An animation of the MCMC sampling scheme in the fully-unsupervised framework can be found online at http://mathsci.ucd.ie/~fmaire/MCoEM/carlinChib.html.

5.2.4. Remark on the number of components in the mixture

Specifying a relevant number of clusters C in situations where the prior knowledge on the data is limited is an arduous task. Processing the data online makes the challenge even bigger since it is not possible to provide an initial meaningful guess on C at the start of the algorithm as data are simply unavailable. Updating C on the fly requires a specific care as some clusters might not be updated in a series of observations and still be relevant to describe future data, as opposed to batch algorithms approaches that discard those clusters that are not frequently updated (see for example Wu and Hitchcock, 2016 where a dynamic rule to update C is suggested). Nevertheless, in the context of the deformable template model, inference carried out by the MCoEM turns out to be rather robust to the choice of C, provided that the specified deformation model is relevant. In particular, our experiments show that when C is increased, the estimated geometric deformation amplitude diminishes. Hence, adding more clusters is equivalent to allow more diversity in the photometry (more templates) while reducing the geometrical variability within each class. In this perspective, C yields a tradeoff on how the variability of the dataset is modeled: low photometric/large geometric variations if C is moderate and conversely. Whether it is accounted as a photometric or a geometric variation, the variability of the data is reasonably well captured by the model: we have checked

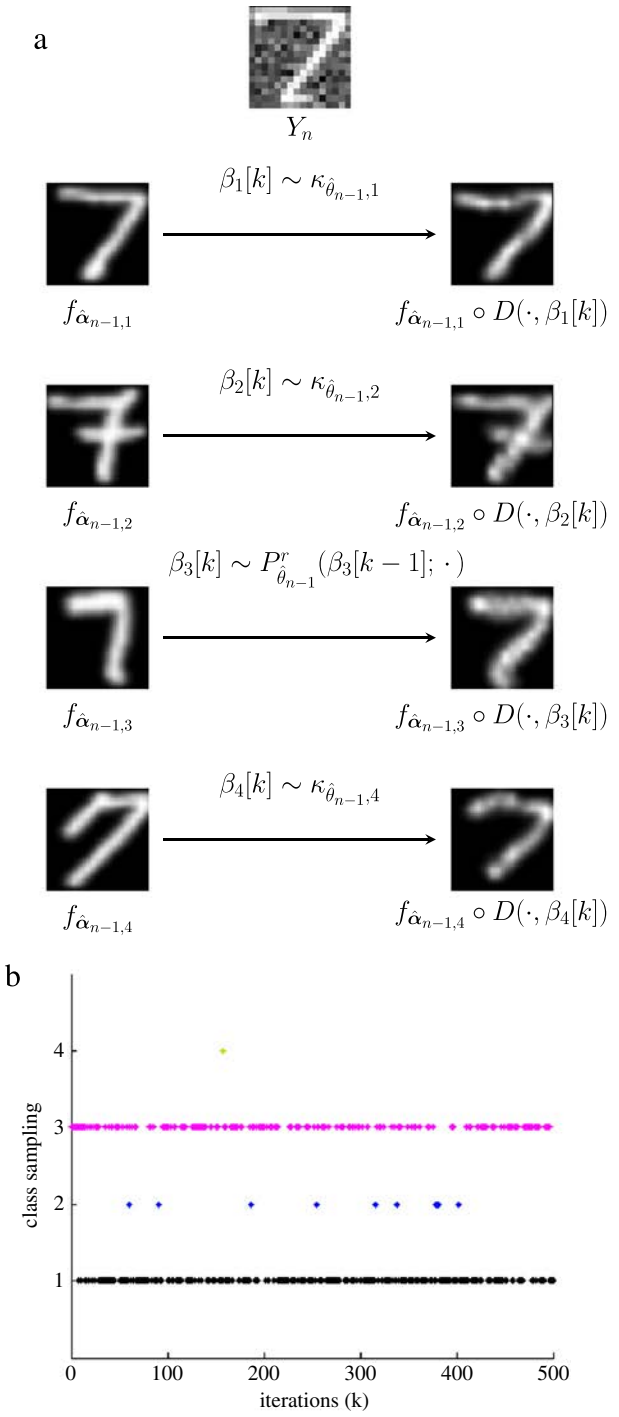


Fig. 8. Sampling missing data $(I[k], \beta_1[k], \ldots, \beta_4[k]) \sim \tilde{\pi}_{\theta_n}(\cdot \mid \mathbf{Y}_n)$ with $n = 600$, using the Carlin and Chib approach introduced in Section 4. The top panel (a) illustrates the sampling of deformations parameters taking place at the $k = 450$th iteration of the Markov chain. The bottom panel (b) illustrates the class index sampled by the Markov chain.

that the log-likelihood at convergence is relatively steady with C. Obviously, the computational cost generated when C is large is prohibitive and, on this example, C needs not be large to perform an accurate inference.

5.2.5. Comparison with SAEM-MCMC

For conciseness, we will write from now on SAEM instead of SAEM-MCMC for the algorithm formalized by Kuhn and Lavielle (2004) and applied to perform template estimation in Allassonnière and Kuhn (2010). Templates estimated by

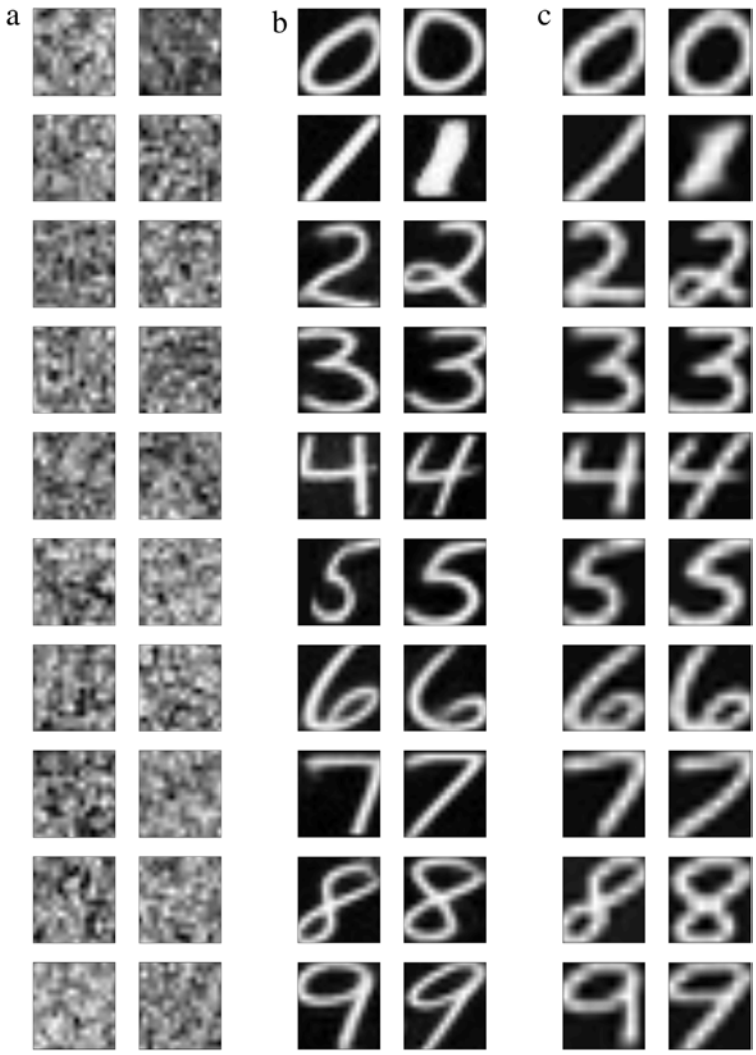


Fig. 9. Templates extracted by MCoEM (b) and SAEM (c) from the same dataset, consisting of $n = 300$ handwritten digit images from each type of digit, in a partially-supervised way and during a 10-hour running time experiment. Each model comprises $C = 2$ classes. (a) represents the initial templates drawn at random $\hat{\boldsymbol{\alpha}}_{0,j} \sim \mathcal{N}(\mathbf{0}_m, M^{-1})$ where M is a square matrix of size m with elements $M_{p,q} = \exp(-\|r_p - r_q\|^2/\nu^2)$.

MCoEM are compared with those obtained by applying SAEM to the same images, in both setups (partially and fully-unsupervised). In the partially-supervised setup, both algorithms processed the same $n = 300$ images for each class of digit, during a 10-hour runtime. In the fully-unsupervised approach, MCoEM and SAEM processed the same $n = 500$ images (50 images of each digit), during a 40-hour runtime experiment. SAEM is a batch stochastic EM algorithm that processes all the data at each iteration. In the mixture of deformable models context, this means that SAEM has to register each single observation with the set of templates estimated at each iteration, whereby generating a significant computational burden. As a consequence, in a 10-hour running time experiment, SAEM could only perform 23 iterations while MCoEM completed nearly 2000 iterations. Figs. 9 and 10 report the sets of templates extracted by both methods in the two setups.

In the partially-supervised setup, the two sets of estimated templates show similar features (Fig. 9), highlighting that in spite of processing the data on the fly, MCoEM yields a similar stability than SAEM. From a qualitative perspective, performing nearly ten times as many iterations than SAEM is beneficial for MCoEM whose templates look much smoother and yield a better resolution. An animation of the template estimation in this setup can be found online at http://mathsci.ucd.ie/~fmaire/MCoEM/templates.html.

The templates estimated by MCoEM and SAEM in the fully-unsupervised setup, implemented with $C = 15$ components, are reported in Fig. 10. The first ten templates are consistent for both algorithms while the last five templates differ significantly. On the one hand, MCoEM only makes use of 13 from the 15 available classes. The two remaining classes corresponds to the 12th and 15th templates in the middle column of Fig. 10. Fig. 11 plots the weight evolution for each class as MCoEM moves forward and shows that those two classes have quickly become unused by the algorithm. The first ten classes weight is slightly lower than 1/10 which is in line with the dataset. On the other hand, SAEM maintains the 15

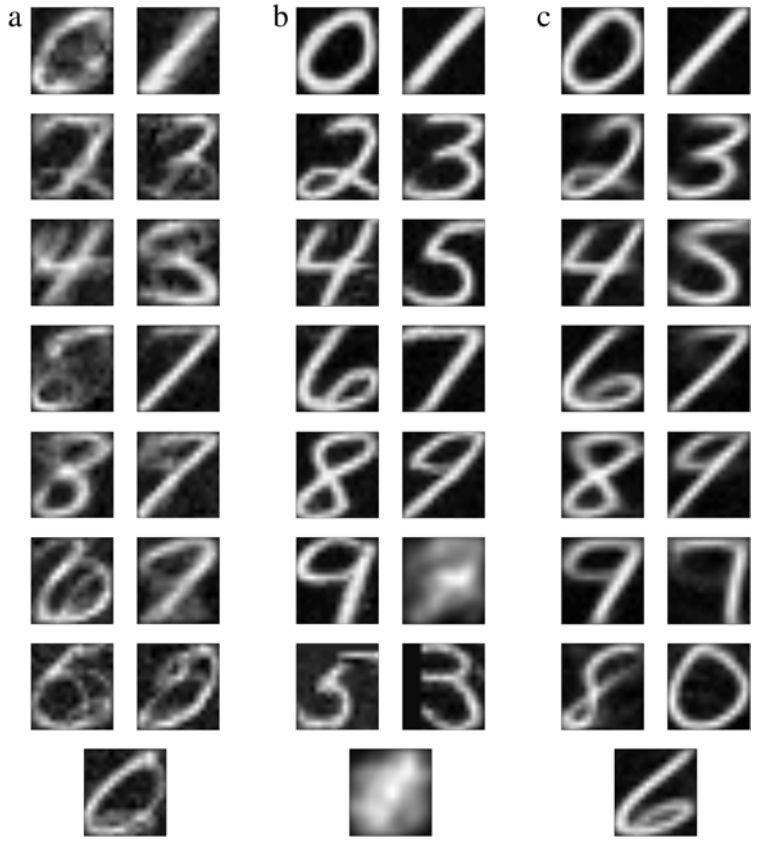


Fig. 10. Templates extracted by MCoEM (b) and SAEM (c) from the same dataset, consisting of $n = 500$ handwritten digit images from each type of digit, in a fully-supervised way and during a 40-hour running time experiment. The model comprises $C = 15$ mixture components. (a) represents the initial templates based on k-means clustering applied on 50 random images. See http://mathsci.ucd.ie/~fmaire/MCoEM/templates.html for an animation.

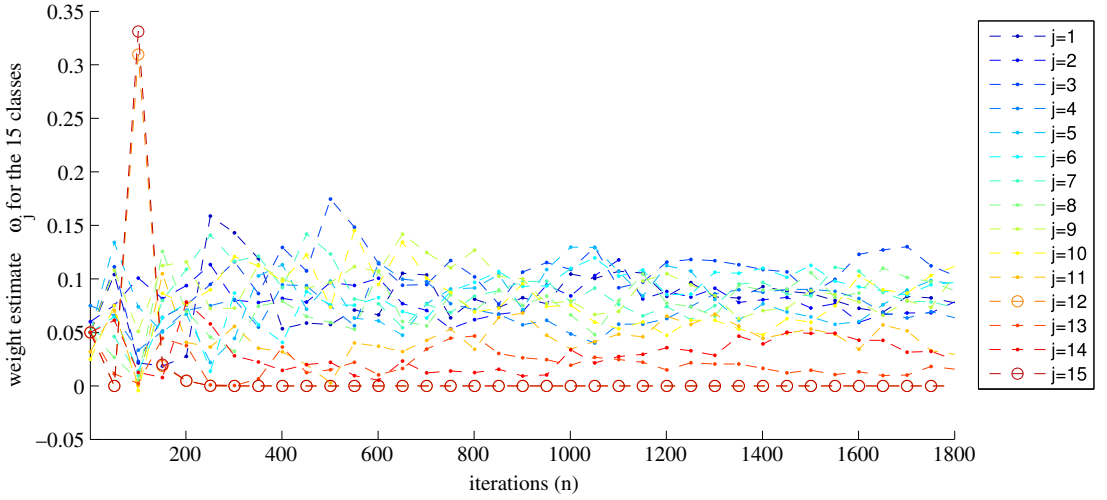


Fig. 11. Evolution of the weight for each of the $C = 15$ classes of the fully-unsupervised mixture of template model inferred by MCoEM. Circled data points correspond to the two classes whose weight vanishes.

classes alive all throughout the algorithm. In this example, SAEM appears more robust than MCoEM for inferring a mixture model. However, we believe that the stability of MCoEM can be improved by increasing the number of iterations before the first parameter update (only 50 in our simulation), hence avoiding this degeneracy problem. Indeed, from Fig. 11 it is clear that those two classes have been left empty after the first 50 iterations, paving the way to the pathological effect observed at the next updates.

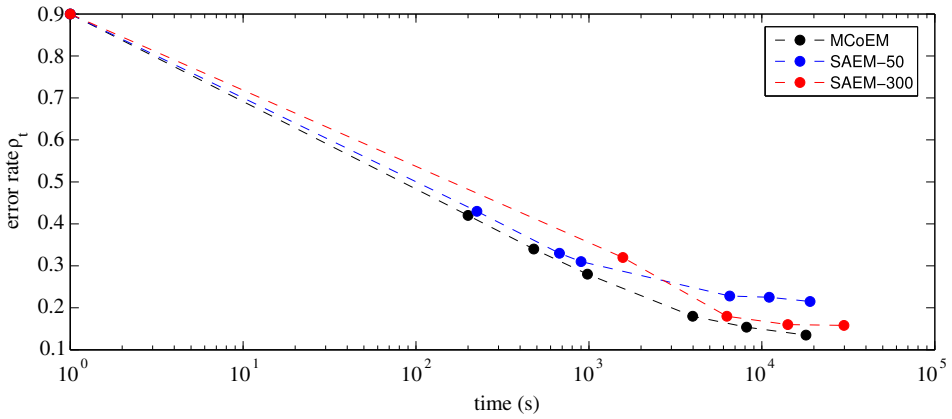


Fig. 12. Live error rate for MCoEM, SAEM-50 and SAEM-300, applied in the partially-supervised setup. For each algorithm, a ball at time t represents the error rate at time t. ρ_t is obtained by comparing the estimated class $\hat{V}_{k,t}$ with the label V_k for the $N = 1000$ testing data. $\hat{V}_{k,t}$ is returned by the classifier making use of the knowledge acquired by the algorithm up to time t. Dashed lines are used for readability only and do not convey any error rate outside the balls.

6. Classification

When considering real-time classification applications, the MCoEM methodology may prove more adequate than SAEM: indeed as soon as the first estimate of θ is available, a classifier can be implemented. Of course, the rate of correct classification is expected to improve upon random guessing as soon as the templates (and the other parameters) take shape. Both algorithms produce a sequence of parameter estimates. However, since iterations of MCoEM and SAEM have different computational complexity, we consider $\hat{\theta}_t$, the parameter estimate after a runtime of t time units, as a fair way to compare both methods.

Learning parameters of the mixture of deformable models (5) allow one to classify labeled observations $\{(\tilde{\mathbf{Y}}_1, V_1), \ldots, (\tilde{\mathbf{Y}}_N, V_N)\}$ gathered in a testing dataset. There is no overlap between those testing observations and the data $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_n\}$ processed by the algorithms during the learning phase. Let ρ_t be the *live* error rate at time t, defined as the empirical rate of uncorrect classification (based on $N = 1000$ testing observations) obtained using the parameters estimated by the algorithms at time t:

$$\rho_t = \frac{1}{N} \sum_{k=1}^{N} \mathbb{1}_{\hat{V}_{k,t} \neq V_k},$$

where $\hat{V}_{k,t}$ is the class of digit assigned to $\tilde{\mathbf{Y}}_k$ returned by the classifier using the estimate θ_t. In this section, we compare the live error rate on the handwritten digits example such that $V_k \in \{0, \ldots, 9\}$ (see Section 5.2) based on estimates from MCoEM (processing a new observation at each iteration), SAEM-50 and SAEM-300, *i.e.* SAEM using $n = 50$ and $n = 300$ learning observations respectively. Both learning setups partially-supervised and fully-unsupervised are considered.

6.1. Partially-supervised learning

In this approach, each type of digit $v \in \{0, \ldots, 9\}$ is described at time t by a set of parameters $(\hat{\theta}_{1,t}^{(v)}, \ldots, \hat{\theta}_{C,t}^{(v)})$. We used $C = 2$ classes per digit in this implementation. The following unnormalized probabilities

$$\text{for all } v \in \{0, \ldots, 9\}, \quad \pi_v(\tilde{\mathbf{Y}}_k, \hat{\theta}_t) = \sum_{i=1}^{C} \mathbb{E}_{\hat{\theta}_{i,t}^{(v)}} \left[g_\theta(\tilde{\mathbf{Y}}_k \mid I_k, \mathbf{X}_k) \mid \tilde{\mathbf{Y}}_k, I_k = i \right], \tag{31}$$

are calculated and the guess $\hat{V}_{k,t}$ is defined as

$$\hat{V}_{k,t}(\hat{\theta}_t) = \arg \max_{v \in \{0,\ldots,9\}} \pi_v(\tilde{\mathbf{Y}}_k, \hat{\theta}_t). \tag{32}$$

The conditional expectation in (32) is intractable and approximated by the sample mean of a Metropolis–Hastings Markov chain targeting the posterior distribution of $\mathbf{X}_k, \pi(\cdot \mid \mathbf{Y}_k, I_k = i)$.

Fig. 12 reports the live error rate for the three algorithms applied in the partially-supervised setup. For each algorithm the error rate at $t = 0$ is $\rho_0 = 0.9$ since the initial template is uninformative (see Fig. 9(a)) The second left-most ball corresponds to the error rate using the first estimate produced by each algorithm. For each algorithm, the time per iteration is reported in Table 1. Note that for numerical stability, MCoEM first parameter update occurred after 15 iterations, the second after 10 iterations and at each iteration from the 20th iteration onwards. SAEM yields a large error rate when learning from only

Table 1
CPU time of an iteration of MCoEM and SAEM with $n = 50$ and $n = 300$ and $C = 2$ mixture components.

	MCoEM	SAEM-50	SAEM-300
CPU time/iteration (s)	20	225	1570

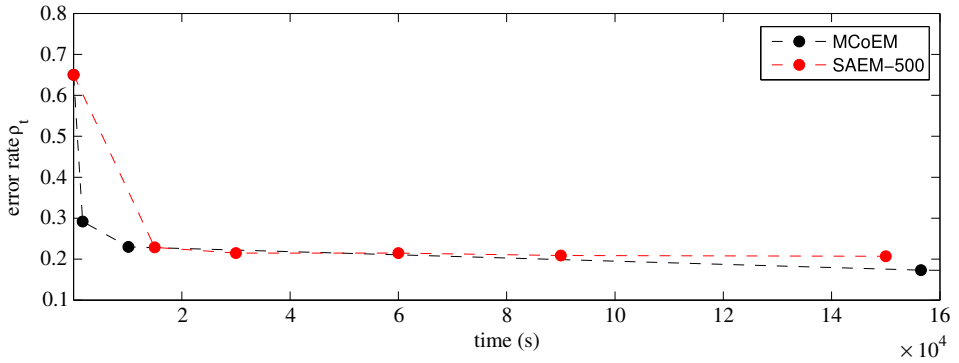


Fig. 13. Live error rate for MCoEM and SAEM-500, applied in the fully-unsupervised setup. Dashed lines are used for readability only and do not convey any error rate outside the balls.

$n = 50$ observations. It is significantly reduced when using $n = 300$ observations but this improvement comes at the price of a prohibitively large computational cost. The first estimate produced by SAEM-300 is available after $t_1 = 1570$ s. At this time, MCoEM has already performed nearly 80 iterations and exhibits a significantly lower live error rate: approximately $\rho_{t_1} = 0.24$ for MCOEM and $\rho_{t_1} = 0.34$ for SAEM-300. MCoEM yields a successful tradeoff between SAEM-50 with limited n allowing quick estimates but poor error rate and SAEM-300 with larger n allowing lower error rate but a slower estimation. In addition, since MCoEM makes use of new data at each iteration, its live error rate is expected to keep reducing while SAEM's error rate, using a fixed dataset, seems to flatten once convergence of the parameter is reached.

6.2. Fully-unsupervised learning

Since in this learning setup, an object (*e.g.* a digit nine) may be described by several templates (3 in the simulation of Fig. 6(c) and 2 in that of Fig. 10(b)), an intermediate layer of classes is required. Based on this observation, an external agent must specify the mapping $M : \mathbb{I} \to \{0, 1, \ldots, 9\}$ that links each class designed by MCoEM/SAEM to the object it describes. Classification is then carried out in the same way as in the semi-supervised setup. More precisely, given the estimate $\hat{\theta}_t$, the following unnormalized probabilities

$$\text{for all } v \in \{0, \ldots, 9\}, \quad \pi_v(\tilde{\mathbf{Y}}_k, \hat{\theta}_t) = \sum_{i \in M(v)} \mathbb{E}_{\hat{\theta}_{i,t}} \left[g_\theta(\tilde{\mathbf{Y}}_k \mid I_k, \mathbf{X}_k) \mid \tilde{\mathbf{Y}}_k, I_k = i \right], \tag{33}$$

are approximated by an MCMC estimate and the guess $\hat{V}_{k,t}$ is derived as in (32).

Fig. 13 reports the live error rate of MCoEM and SAEM-500 in the fully-unsupervised setup. At time $t = 0$, the error rate ρ_0 is 0.65 and not 0.9 as in the previous setup. This is because the initial templates are derived from k-means centroids based on the same $n = 50$ data (see Section 5.2.2) and are thus no longer non-informative. SAEM is clearly penalized by processing $n = 500$ observations and estimating $C = 15$ classes of parameters: it nearly takes 5 h of computation to get the first SAEM's estimate. Interestingly, SAEM's first estimate is nearly as "good", in the error rate sense, as the MCoEM estimate obtained after 5 hours. Nevertheless, using MCoEM offers a practitioner the possibility to classify much quicker new observations (see Table 2 that compares the computational cost of one iteration of MCoEM and SAEM).

7. Discussion

We have proposed a statistical framework to perform sequential and unsupervised inference of a deformable template model, with application to curve synchronization and shape extraction. It makes use of the Monte Carlo online EM algorithm (MCoEM), derived from Cappé and Moulines (2009) and a novel MCMC sampling method, based on the Carlin and Chib sampler (Carlin and Chib, 1995), allowing to simulate the unsamplable joint distribution of the cluster index and deformation parameters. The method has been applied successfully to extract reference templates from several datasets featuring high time/geometric dispersion.

Table 2
CPU time of an iteration of MCoEM and SAEM $n = 500$ and $C = 15$ mixture components.

	MCoEM	SAEM-500
CPU time/iteration (s)	170	17,570

Our work was primarily motivated by the computational gain arising when processing one observation at a time. Indeed, when the missing data is a large vector and many observations are available, stochastic batch EM algorithms such as SAEM (Delyon et al., 1999) are prohibitively slow for practical use. This has been illustrated with the classification problem (Section 6.2) in which SAEM's error rate after nearly 5 h of computation is still at the initial level. In comparison, it took MCoEM less than 20 min to reach less than half the initial error rate. In this perspective, MCoEM can be regarded as a linearization of stochastic batch EM algorithms, which is particularly appealing in a Big Data context. Note that SAEM coupled with an efficient MCMC kernel such as the Anisotropic Metropolis Adjusted Langevin sampler (AMALA) is likely to speed up the E-step approximation, as less MCMC transitions is required. This has been successfully demonstrated in Allassonniere and Kuhn (2015) in a similar context and the adaptation of their methodology to handle a mixture of deformable templates could, from a computational viewpoint, compete with our online approach.

In terms of implementation, the main concern when inferring a mixture model with MCoEM is class degeneracy. To mitigate this risk, two points have been discussed. First, a particular care should be brought to the way initial parameters are set and especially the templates. We have suggested to use k-means clustering on a limited set of observations to initiate the templates. Second, the number of EM iterations between the first parameter updates should be large enough in order to assign at least one observation to each class. Adaptive implementations have not been considered but could yield an automated update schedule.

The handwritten digit example studied in this paper shows that MCoEM seems to inherit SAEM's asymptotic behavior. Indeed, (i) qualitatively, the template shapes extracted by both algorithms are similar and (ii) quantitatively, the error rates are comparable. This result calls for further investigation as a theoretical framework is yet to be developed to establish the convergence of MCoEM. Both SAEM and online EM proofs of convergence relies on stochastic approximation theory arguments. However those proofs cannot be straightforwardly extended to MCoEM since it combines two approximations: one on the conditional expectation (which is in SAEM) and the other one on the data generating process (which is in the online EM). We therefore leave this as a future work. Interesting questions involved whether the convergence rate of $n^{-1/2}$ achieved in the online EM (Cappé and Moulines, 2009) is degraded when replacing the expectation of the sufficient statistics by an unbiased estimate and how the variance of the unbiased estimator propagates to the asymptotic variance of the estimator.

Acknowledgments

The authors would like to thank the Editor and two referees for their comments and suggestions which greatly helped to improve the overall presentation of the manuscript. Florian Maire would like to thank the ONERA-The French Aerospace Lab and the DGA-The French Procurement Agency.

References

Allassonnière, S., Amit, Y., Trouvé, A., 2007. Towards a coherent statistical framework for dense deformable template estimation. J. R. Stat. Soc. Ser. B Stat. Methodol. 69 (1), 3–29.
Allassonnière, S., Bigot, J., Glaunès, J.A., Maire, F., Richard, F.J.-P., 2013. Statistical models for deformable templates in image and shape analysis. Ann. Math. Blaise Pascal 20 (1), 1–35.
Allassonnière, S., Kuhn, E., 2010. Stochastic algorithm for parameter estimation for dense deformable template mixture model. ESAIM Probab. Stat. 14, 382–408.
Allassonniere, S., Kuhn, E., 2015. Convergent Stochastic Expectation Maximization algorithm with efficient sampling in high dimension. Application to deformable template model estimation. Comput. Statist. Data Anal. 91, 4–19.
Andrieu, C., De Freitas, N., Doucet, A., Jordan, M.I., 2003. An introduction to MCMC for machine learning. Mach. Learn. 50 (1–2), 5–43.
Bernhardt, P.W., Zhang, D., Wang, H.J., 2015. A fast EM algorithm for fitting joint models of a binary response and multiple longitudinal covariates subject to detection limits. Comput. Statist. Data Anal. 85, 37–53.
Bigot, J., Charlier, B., 2011. On the consistency of fréchet means in deformable models for curve and image analysis. Electron. J. Stat. 5, 1054–1089.
Cappé, O., Moulines, E., 2009. Online Expectation–Maximization algorithm for latent data models. J. R. Stat. Soc. Ser. B Stat. Methodol. 71 (3), 593–613.
Carlin, B.P., Chib, S., 1995. Bayesian model choice via Markov chain Monte Carlo. J. R. Stat. Soc. Ser. B Stat. Methodol. 57, 473–484.
Castellanos, N.P., Angel, P.L.D., Medina, V., 2004. Nonrigid medical image registration technique as a composition of local warpings. Pattern Recognit. 37 (11), 2141–2154.
Christensen, G., 1999. Consistent linear-elastic transformations for image matching. In: Information Processing in Medical Imaging. pp. 224–237.
Christensen, G., Rabbitt, R., Miller, M., 1996. Deformable templates using large deformation kinematics. IEEE Trans. Image Process. 5 (10), 1435–1447.
Delyon, B., Lavielle, M., Moulines, E., 1999. Convergence of a stochastic approximation version of the EM algorithm. Ann. Statist. 27, 94–128.
Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1), 1–38. (with discussion).
Dimeglio, C., Gallón, S., Loubes, J.-M., Maza, E., 2014. A robust algorithm for template curve estimation based on manifold embedding. Comput. Statist. Data Anal. 70, 373–386.
Frey, B.J., Jojic, N., 2003. Transformation-invariant clustering using the EM alogrithm. IEEE Trans. Pattern Anal. Mach. Intell. 25 (1), 1–17.
Gaffney, S., Smyth, P., 2004. Joint probabilistic curve clustering and alignment. In: Advances in Neural Information Processing Systems 17.
Gelfand, A.E., Smith, A.F.M., 1990. Sampling based approaches to calculating marginal densities. J. Amer. Statist. Assoc. 85, 398–409.

Geman, S., Geman, D., 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. 6, 721–741.
Hull, J.J., 1994. A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Mach. Intell. 16 (5), 550–554.
Kneip, A., Gasser, T., 1992. Statistical tools to analyze data representing a sample of curves. Ann. Statist. 20 (3), 1266–1305.
Kuhn, E., Lavielle, M., 2004. Coupling a stochastic approximation version of EM with an MCMC procedure. ESAIM Probab. Stat. 8, 115–131.
Liu, Z., Almhana, J., Choulakian, V., McGorman, R., 2006. Online EM algorithm for mixture with application to Internet traffic modeling. Comput. Statist. Data Anal. 50 (4), 1052–1071.
Liu, X., Yang, M., 2009. Simultaneous curve registration and clustering for functional data. Comput. Statist. Data Anal. 53 (4), 1361–1376.
Ma, J., Miller, M.I., Trouvé, A., Younes, L., 2008. Bayesian template estimation in computational anatomy. NeuroImage 42 (1), 252.
McLachlan, G.J., Krishnan, T., 2007. The EM Algorithm and Extensions. Vol. 382. Wiley-Interscience.
Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equations of state calculations by fast computing machines. J. Chem. Phys. 21, 1087–1092.
Moffa, G., Kuipers, J., 2014. Sequential Monte Carlo EM for multivariate probit models. Comput. Statist. Data Anal. 72, 252–272.
Nguyen, H.D., McLachlan, G.J., Wood, I.A., 2016. Mixtures of spatial spline regressions for clustering and classification. Comput. Statist. Data Anal. 93, 76–85.
Ramsay, J.O., 2006. Functional Data Analysis. Wiley Online Library.
Ramsay, J.O., 2011. Curve registration. In: The Oxford Handbook of Functional Data Analysis. pp. 235–258.
Ramsay, J.O., Li, X., 1998. Curve registration. J. R. Stat. Soc. Ser. B Stat. Methodol. 60 (2), 351–363.
Silverman, B.W., 1985. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. J. R. Stat. Soc. Ser. B Stat. Methodol. 47, 1–52.
Telesca, D., Inoue, L., 2008. Bayesian hierarchical curve registration. J. Amer. Statist. Assoc. 103, 328–339.
Tuddenham, R., Snyder, M.M., 1954. Physical growth of california boys and girls from birth to eighteen years. Publications in child development. University of California, Berkeley 1 (2), 183.
Wang, K., Gasser, T., 1997. Alignment of curves by dynamic time warping. Ann. Statist. 25, 1251–1276.
Wolfinger, R., 1993. Laplace's approximation for nonlinear mixed models. Biometrika 80 (4), 791–795.
Wu, Z., Hitchcock, D.B., 2016. A Bayesian method for simultaneous registration and clustering of functional observations. Comput. Statist. Data Anal. 101, 121–136.
Zhong, Z., 2008. Curve registration in functional data analysis. ProQuest.