Estimating Random-Intercept Models on Data Streams

L. Ippel[a,*], M.C. Kaptein[a], J.K. Vermunt[a]

[a] *Tilburg University, Warandelaan 2, PObox 90153, 5000LE Tilburg, the Netherlands*

Abstract

Multilevel models are often used for the analysis of grouped data. Grouped data occur for instance when estimating the performance of pupils nested within schools or analyzing multiple observations nested within individuals. Currently, multilevel models are mostly fit to static datasets. However, recent technological advances in the measurement of social phenomena have led to data arriving in a continuous fashion (i.e., data streams). In these situations the data collection is never "finished". Traditional methods of fitting multilevel models are ill-suited for the analysis of data streams because of their computational complexity. A novel algorithm for estimating random-intercept models is introduced. The Streaming EM Approximation (SEMA) algorithm is a fully-online (row-by-row) method enabling computationally-efficient estimation of random-intercept models. SEMA is tested in two simulation studies, and applied to longitudinal data regarding individuals' happiness collected continuously using smart phones. SEMA shows competitive statistical performance to existing static approaches, but with large computational benefits. The introduction of this method allows researchers to broaden the scope of their research, by using data streams.

Keywords: Data streams, Expectation-Maximization algorithm, Multilevel Models, Online learning, Random-Intercept model

1. Introduction

In the social sciences we often encounter grouped data, such as pupils grouped within school classes (e.g., Barrett et al., 2013), multiple observations grouped within individuals (Killingsworth and Gilbert, 2010), or voters grouped within geographical regions (Gelman, 2007). Such data are typically analyzed using multilevel (or hierarchical) models in which batches of group-level parameters are

*Corresponding author
Lianne Ippel, G.J.E.Ippel@tilburguniversity.edu
Online supplementary material is available.

treated as randomly drawn from an underlying distribution. In this paper we will use the formulation of "observations nested within individuals", although the method we present does not restrict itself to this type of nesting.

Multilevel models have various advantages over more traditional methods of analysis, such as aggregated analysis, in which the within-group structure is ignored, or group-specific analysis, in which information about the other groups is ignored. That is, they

1. contain fewer parameters than group-specific models,
2. allow for generalization of results to a wider population of groups, and
3. allow information to be shared between groups (Raudenbush and Bryk, 2002; Steenbergen and Jones, 2002).

The latter property in particular makes multilevel analysis interesting when the focus is on obtaining group-level predictions, since multilevel modeling yields smaller out-of-sample prediction error than predictions derived from either an aggregate or a group-specific analysis (see e.g., Morris and Lysy, 2012).

Current (maximum-likelihood) methods for fitting multilevel models use iterative algorithms such as Newton-Raphson or Expectation Maximization (EM, Dempster, Laird, and Rubin, 1977) to maximize the likelihood. Alternatively, but not considered in this paper, one could use a Bayesian framework with MCMC sampling (for more details see, e.g., Browne and Goldstein, 2010). However, each of these methods require multiple passes through the full dataset to obtain parameter estimates. Even though fitting a multilevel model once, on a moderately sized dataset does often not require excessive computation time, such ways of fitting multilevel models can become infeasible when a dataset is extremely large, or in the situation where the data collection is never "finished" because more data present themselves over time.

Recent technological developments have, however, led to the increased availability of these so-called data streams: i.e., datasets which are continuously augmented with new data points. Such data streams often have a grouped (or nested) structure. Examples include fraud detection using credit card transactions, where transactions are nested within credit cards (Patidar and Sharma, 2011), telephone communication analysis, where calls are nested within telephone registrations (Cortes, Fisher, Pregibon, Rogers, and Smith, 2000), and consumer behavior tracking in e-commerce, where purchased items or visited web pages are nested within customers (Lee, Podlaseck, Schonberg, and Hoch, 2001). In order to obtain up-to-date predictions of the individual-level effects, the parameters of the model of interest should be updated as data points come in, and the updated model parameters should be used for prediction purposes. When applied to streaming data, these

traditional methods have to repeatedly cycle through all available data points, each time a new data point arrives, in order to obtain up-to-date parameter estimates. Additionally, even if the dataset is no longer augmented, but static and (extremely) large, it is often computationally preferable to analyze the dataset in smaller batches, or even a data point at a time (Ng and McLachlan, 2003; Thiesson, Meek, and Heckerman, 2001). We propose an adaption of the EM algorithm for the estimation of random-intercept models, to resolve the problem of analyzing grouped data in a data stream or when the dataset is extremely large.

The resulting Streaming EM Approximation algorithm (henceforth referred to as SEMA) falls within the framework of online learning methods (Gaber et al., 2005). A key feature of online learning is that the data are summarized into a few summary statistics which contain all relevant information of previous data points (Opper, 1998). SEMA is an approximate EM method, because unlike the EM algorithm which uses all the data to update the model parameters, we only use a single data point, some summary statistics on the individual level, and the previous estimates of the model parameters, to update the model parameters. Because SEMA does not require all the data to be in memory, SEMA is more appropriate to deal with data streams than the conventional EM algorithm.

Related methods for speeding up the EM algorithm have been proposed for dealing with large (static) datasets, for example, Berlinet and Roland (2012) discussed methods to speed up the convergence rate of the conventional EM algorithm. Wolfe, Haghighi, and Klein (2008) presented an (offline) parallel version of the EM algorithm and McLachlan and Peel (2000, ch. 12) described various possible adaptations of EM methods for large datasets. Various online adaptations of the EM algorithm for different applications have also been proposed, for example, for mixture models (see, e.g., Cappé and Moulines, 2009; Liu, Almhana, Choulakian, and McGorman, 2006; McLachlan and Peel, 2000; Wolfe et al., 2008) and for latent variable models (Cappé and Moulines, 2009). Instead of speeding up the EM algorithm, Steiner and Hudec (2007) proposed a method to scale down the data prior to using the EM algorithm. We add to this existing literature by proposing an EM approximation for the estimation of models based on data streams consisting of *dependent* observations. The method we propose stores information on the level of individuals, instead of the level of observations, and updates the estimates in a single pass over the data, making it suitable for both data streams and extremely large datasets.

The remainder of this article is organized as follows. In the next section, we illustrate the computational advantages of streaming estimation using the simple example of the estimation of a sample mean. Next, we discuss the estimation of random-intercept models using the EM algorithm, and show how this algorithm can be modified into a streaming version, leading to SEMA. Subsequently we eval-

uate SEMA in two simulation studies. In the first simulation study we evaluate the accuracy of the estimates of the model parameters, and of the individual-level effects. In the second study we evaluate three alternative implementations of SEMA to improve the estimates both of the model parameters and of the individual-level effects. The first alternative uses a small part of the data to obtain better starting values, the second implementation cycles through all individuals at given intervals, and the last implementation is a combination of the previous two. In Section 5 we illustrate the use of SEMA in an application using real data on respondents' happiness, in which nested data, collected using a smart-phone application, "arrived" in a stream. In Section 6 we detail some theoretical characteristics of SEMA, and we discuss a convergence diagnostic to evaluate the estimated model parameters of SEMA. In the following section, we extend the random-intercept model to include additional fixed covariates. The last section discusses the main results of the simulation studies and presents directions for future work.

2. From offline to online data analysis

Before introducing SEMA, we first explain the key changes involved when moving from the *offline* analysis of static datasets to the *online* analysis of data streams. This conceptual shift is easily illustrated by examining the computation of a sample mean $\bar{x}_n$. The standard offline computation proceeds as follows:

$$\bar{x}_n = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{1}$$

where x_i denotes the measurement for the ith unit and n the total number of observations.

Suppose now that we want to compute the sample mean *and* that data enter in a stream. The naive application of the above offline formula would then imply that each time a new data point enters one has to count the number of observations n and compute the sum of all measurements x_i. This is feasible as long as n is not too large or when the update is only required rarely. However, even a simple computation as in Equation 1 becomes infeasible when it needs to be performed in the face of rapidly entering data points, as n grows larger and larger.

The online computation of a sample mean can be done by noting that the sample mean for $n+1$ data points can be expressed as an *update* of the estimated

sample mean for n data points $x_1, \ldots, x_n$. More specifically,

$$\begin{aligned} \bar{x}_{n+1} &= \frac{\sum_{i=1}^{n+1} x_i}{n+1}, \\ &= \bar{x}_n \frac{n}{n+1} + \frac{x_{n+1}}{n+1}, \\ &= \bar{x}_n + \frac{x_{n+1} - \bar{x}_n}{n+1}. \end{aligned} \tag{2}$$

The last line of Equation 2 shows two key features of online learning: first, when a new observation enters, we update the current estimate without revisiting all the historical data. This reduces the computational complexity required to update the sample mean. Note that the number of (offline) computations needed to compute the sample mean as n grows, progresses as follows:

$$\begin{aligned} 1 + 2 + 3 + \cdots + n &= \frac{1}{2} n(n+1), \\ &= O(n^2). \end{aligned}$$

In comparison, the online update of the sample mean, requires the following number of computations

$$\begin{aligned} 1 + 1 + 1 + \cdots + 1 &= n, \\ &= O(n). \end{aligned}$$

This simple analysis shows that the computations to update the mean offline grow quadratically in n, while online the number grows linearly as a function of n.

Second, only certain summary statistics (here n and $\bar{x}_n$) are kept in memory. This makes online learning both computationally fast as well as memory efficient. Similar algorithms can be used, amongst others, for updating of higher moments (Welford, 1962) or for estimating the coefficients of a linear regression model using least squares (Escobar and Moser, 1993; Plackett, 1950). In the next section we detail the transition from offline estimation to online estimation of the random-intercept model.

3. Online estimation of random-intercept models

3.1. The random-intercept model and its standard offline estimation

In this section we will describe the random-intercept model with continuous outcomes, which we focus on throughout this paper. Next, we will give a conceptual

description of the EM algorithm, as well as the technical details of fitting the random-intercept model. These technical details are subsequently needed to explain the transition from offline estimation of the random-intercept model to the online estimation of this model.

The model of interest can be formulated as follows:

$$y_{ij} = \mu_j + \epsilon_{ij}, \quad i = 1 \dots n_j,\ j = 1 \dots J, \tag{3}$$

where y_{ij} is observation i of individual j, n_j is the total number of observations of an individual, J is the total number of individuals, and μ_j is the individual-level random intercept. These intercepts are assumed to be normally distributed as $\mu_j \sim \mathcal{N}(\mu, \tau^2)$. The random error per observation is denoted by ϵ_{ij} and is also assumed to be normally distributed $\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ and independent of μ_j. The three unknown model parameters to be estimated are thus μ, τ^2, and σ^2.

Maximum-likelihood estimates of the parameters of the random-intercept model cannot be computed directly due to the fact that μ_j is not observed. In order to obtain maximum-likelihood estimates, we use the Expectation Maximization algorithm (Dempster et al., 1977). The EM algorithm uses the complete-data log-likelihood function, a likelihood function for which the latent variable (μ_j) is assumed to be known. The complete-data log-likelihood function is as follows:

$$\begin{aligned} \ell(\mu, \tau^2, \sigma^2 | y) = &-\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\sigma^2 - \frac{1}{2}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\frac{(y_{ij}-\mu_j)^2}{\sigma^2} \\ &-\frac{J}{2}\ln(2\pi) - \frac{J}{2}\ln\tau^2 - \frac{1}{2}\sum_{j=1}^{J}\frac{(\mu_j-\mu)^2}{\tau^2}, \end{aligned} \tag{4}$$

where $n = \sum_{j=1}^{J} n_j$.

Because μ_j is not observed, we have to impute values for this variable in order to compute the Complete Data Sufficient Statistics (CDSS). There are three CDSS, one for each model parameter. We denote these CDSS for μ, τ^2, and σ^2 by T_1, T_2, and T_3, respectively. In order to compute the CDSS, the algorithm imputes values for the latent variable in the E step. Using these imputed values in combination with the model parameters of the previous iteration (or starting values) the CDSS are computed. Subsequently, these CDSS are used in the M step. The M step maximizes the complete-data log-likelihood (Eq. 4), given the CDSS of the previous E step.

The CDSS computed in the E step are a function of three individual-level parameters. These individual-level parameters are a function of the observations of individual j and the estimates of the model parameter at iteration $k-1$. These individual-level parameters are $\hat{\mu}_{j(k)}$, $\hat{\rho}_{j(k)}$, and $\hat{\nu}_{j(k)}$, which represent the

individual-level effect, the reliability of this individual-level effect, and variance of this individual-level effect respectively, at iteration k.

We can obtain, $\hat{\mu}_{j(k)}$ using

$$\hat{\mu}_{j(k)} = \hat{\rho}_{j(k)}\bar{y}_j + (1 - \hat{\rho}_{j(k)})\hat{\mu}_{(k-1)}, \tag{5}$$

where $\bar{y}_j$ is the individual average, and where $\hat{\rho}_j$ equals

$$\hat{\rho}_{j(k)} = \frac{\hat{\tau}^2_{(k-1)}}{\hat{\tau}^2_{(k-1)} + \hat{\sigma}^2_{(k-1)}/n_j}. \tag{6}$$

Note that one minus the reliability, $\hat{\rho}_{j(k)}$, can be interpreted as a *shrinkage factor*, which determines the extent to which the estimated individual-level effect, $\hat{\mu}_j$, is moved towards the overall mean, $\hat{\mu}$, (see for instance, Morris and Lysy, 2012; Stein, 1956). When τ^2 is large compared to the residual variance, σ^2, the reliability goes up. The reliability also goes up when the number of observations per individual, n_j, increases. Lastly we compute

$$\hat{\nu}_{j(k)} = \hat{\tau}^2_{(k-1)}(1 - \hat{\rho}_{j(k)}), \tag{7}$$

which can be interpreted as a measure of uncertainty of the individual-level effect.

The CDSS are then computed as follows:

$$T_{1(k)} = \sum_{j=1}^{J} \hat{\mu}_{j(k)}, \tag{8}$$

$$T_{2(k)} = \sum_{j=1}^{J} (\hat{\mu}^2_{j(k)} + \hat{\nu}_{j(k)}), \tag{9}$$

$$T_{3(k)} = \sum_{j=1}^{J} \sum_{i=1}^{n_j} \left[(y_{ij} - \hat{\mu}_{j(k)})^2 + \hat{\nu}_{j(k)} \right]. \tag{10}$$

In the M step, these CDSS are used to obtain new estimates $\hat{\mu}_{(k)}$, $\hat{\tau}^2_{(k)}$, and $\hat{\sigma}^2_{(k)}$. That is,

$$\hat{\mu}_{(k)} = \frac{T_{1(k)}}{J}, \tag{11}$$

$$\hat{\tau}^2_{(k)} = \frac{T_{2(k)}}{J} - \hat{\mu}^2_{(k)}, \tag{12}$$

$$\hat{\sigma}^2_{(k)} = \frac{T_{3(k)}}{n}. \tag{13}$$

After updating the model parameters a new E step is executed, followed by an M step. This process is repeated until convergence.

3.2. Online estimation of the random-intercept model

For streaming estimation of the random-intercept model, an algorithm is needed that does not require storing all the data in memory, or cycling through all the data points at each iteration cycle. For this purpose we propose a modification of the E step of the EM algorithm described previously. This modification involves updating the contribution to the CDSS *only* for the individual for which a new data point enters. The M step remains the same since, given the CDSS, the M step is independent of the data points.

The key feature used by our proposed SEMA algorithm is that the CDSS T_1, T_2, and T_3 can be computed at the level of individuals instead of observations within individuals. Therefore it is no longer required to store all n observations, we merely store a small number of summaries for each of the J individuals.

Let j_t denote the individual which corresponds to the t-th data point. Note that this data point can be either from an individual who is already in the sample, or from a new individual. For the discussion of SEMA, the iteration index, which was previously denoted by k, is now replaced by t, indexing the data point which is being processed. The key element of the proposed SEMA algorithm is that $\hat{\mu}_j$, $\hat{\rho}_j$, and $\hat{\nu}_j$ are computed only for individual $j = j_t$, that is, the individual for which a new data point arrives. This implies that when going from the CDSS based on $t-1$ data points, denoted by $T_{w(t-1)}$, $w \in \{1, 2, 3\}$ to those based on t data points, $T_{w(t)}$, *only* the contribution of individual $j = j_t$ needs to be updated. This can be expressed as follows:

$$T_{w(t)} = T_{w(t-1)} - T_{wj_t(t-1)} + T_{wj_t(t)}, \tag{14}$$

where $T_{wj_t(t-1)}$ and $T_{wj_t(t)}$ denote the contribution to CDSS for individual j_t before and after the entry of data point t. Note that $T_{wj(t)} = T_{wj(t-1)}$ for $j \neq j_t$; that is, the contribution does not change if the new data point does not concern individual j.

Equation 8 (T_1, CDSS for μ) and Equation 9 (T_2, CDSS for τ^2) are already written as a sums over J individuals instead of data points. Therefore they are easily rewritten in the format of Equation 14:

$$\begin{aligned} T_{1(t)} &= \sum_{j=1}^{J} \hat{\mu}_{j(t)}, \\ &= \sum_{j=1}^{J} T_{1j(t)}, \\ &= T_{1(t-1)} - T_{1j_t(t-1)} + T_{1j_t(t)}. \end{aligned} \tag{15}$$

The difference between Equation 8 and Equation 15 is that in the former all $\hat{\mu}_j$ are estimated with the model parameters from the latest iteration. In the latter

formulation however, only for person j_t, $\hat{\mu}_j$ is computed using the most recent model parameters. Therefore, SEMA applies a *partial* E step, only for 1 individual (see also, McLachlan and Peel, 2000; Neal and Hinton, 1998). We rewrite T_2 in a similar way:

$$\begin{aligned} T_{2(t)} &= \sum_{j=1}^{J} \hat{\mu}_{j(t)}^2 + \hat{\nu}_{j(t)}, \\ &= \sum_{j=1}^{J} T_{2j(t)}, \\ &= T_{2(t-1)} - T_{2j_t(t-1)} + T_{2j_t(t)}. \end{aligned} \tag{16}$$

The update of the CDSS of the residual variance (T_3, Eq. 10) differs from the previous two equations. This is due to the fact that Equation 10 presents a summation over n observations, while here we present the computation more efficiently as a summation over J individuals. However, this is relatively straightforward:

$$\begin{aligned} T_{3(t)} &= \sum_{j=1}^{J} \sum_{i=1}^{n_j} \left[(y_{ij} - \hat{\mu}_{j(t)})^2 + \hat{\nu}_{j(t)} \right], \\ &= \sum_{j=1}^{J} (\bar{y_j^2} - 2\bar{y}_j \hat{\mu}_{j(t)} + \hat{\mu}_{j(t)}^2 + \hat{\nu}_{j(t)}) n_j, \\ &= \sum_{j=1}^{J} T_{3j(t)}, \\ &= T_{3(t-1)} - T_{3j_t(t-1)} + T_{3j_t(t)}, \end{aligned} \tag{17}$$

where $\bar{y_j^2}$ is the average of the squared y_{ij} for individual j. This analysis shows that in order to perform the E step we do not need all data points, but only $\bar{y}_j$, $\bar{y_j^2}$ and n_j.

To summarize, at entry of data point t the SEMA algorithm proceeds as follows:

- E step: for $j = j_t$,
    1. subtract the current contribution from the CDSS,
    2. update $\bar{y}_j$, $\bar{y_j^2}$, and n_j online,
    3. compute new $\hat{\mu}_{j(t)}$, $\hat{\rho}_{j(t)}$, and $\hat{\nu}_{j(t)}$,
    4. add the new contribution to the CDSS,
- M step

1. increase J by 1 when it concerns an observation of a new individual and set $n = t$,
2. compute the new estimates $\hat{\mu}_{(t)}$, $\hat{\tau}^2_{(t)}$, and $\hat{\sigma}^2_{(t)}$ based on the CDSS from the previous E step.

In the summary presented above, it can be seen why SEMA (Streaming EM Approximation) is called an approximate EM algorithm: SEMA performs, like EM, an E step and a M step. However, unlike the EM algorithm, it only does a single partial E step, because it only updates the contribution to the CDSS for a single individual. Doing only an E step for one individual is computationally less expensive than doing the E step for all individuals. It also means that SEMA will converge more slowly (i.e., take more partial E - M steps) to the (local) maximum-likelihood estimate than the EM algorithm, because it only updates the information of one individual instead of the updating the information of all individuals. The benefit of SEMA is that it is computationally less intensive, which makes it suitable for dealing with both very large static data and data streams, because the required memory only grows with the number of individuals instead of the number of data points. An example of the SEMA algorithm in [R] code is available at github.com/L-Ippel/SEMA. In Section 6 we provide some additional justification for SEMA. In the next section we will test the accuracy of SEMA in two simulation studies.

4. Performance of SEMA evaluated by simulation

4.1. Simulation study I: Evaluation of the precision of estimated parameters

4.1.1. Design

In this simulation study, we compare the performance of the proposed SENA algorithm with the standard EM algorithm, in terms of the accuracy of the parameter estimates. An important factor affecting the speed of convergence of the EM algorithm for multilevel models is the average reliability $\bar{\rho}$ (see also Eq. 6); that is, when $\bar{\rho}$ is large, the EM algorithm will converge after a few iterations, but when the ρ_j's get closer to zero convergence will become slower. Since it can be expected that convergence of SEMA will be strongly affected by $\bar{\rho}$, this is the main factor varied in this simulation study. We do so in two different ways: by varying the number of observations per individual, n_j, and by varying the amount of variance of the random intercept, τ^2. In this simulation study we keep the residual variance, σ^2, constant. We evaluate SEMA and EM by monitoring the parameter estimates and predicted individual-level effects during the data stream.

We generated data streams of $n = 10,000$ observations. The average number of observations per individual (n_j) was 10, 25, or 100, which results in J = 1,000, 250, or 100 individuals in total. The individual-level effects, μ_j, were drawn from a normal distribution with $\mu = 10$ and variance $\tau^2 = \{1, 10, 25, 100\}$. The residual variance was set to $\sigma^2 = 100$ in all conditions. First we generated J individual-level effects from $\mu_j \sim \mathcal{N}(\mu, \tau^2)$. Next, the observations were generated by randomly drawing an individual, and generating a data point based on this individual's true individual-level effect. The 12 different settings for n_j and τ^2 yielded average reliabilities $\bar{\rho}$ ranging from .091 to .990. Table 1 presents the different levels of $\bar{\rho}$ in the simulation study.

Each of these 12 conditions was run 1,000 times. The starting values used for the model parameters were $\hat{\mu}_{(0)} = y_{t=1}$ (first observation of the data stream), $\hat{\tau}^2_{(0)} = 1$, and $\hat{\sigma}^2_{(0)} = 1$. Both the simulation of the data stream, and the estimation using SEMA and EM, were implemented in [R](R Core Team, 2013).

4.1.2. Results

Tables 2 through 5 present the mean and standard deviation (SD) of $\hat{\mu}$, $\hat{\sigma}^2$, and $\hat{\tau^2}$ respectively, and the averaged squared prediction error: $\bar{e^2} = \sum_{j=1}^{J} \frac{(\hat{\mu}_j - \mu_j)^2}{J}$ across 1,000 replications at 100, 1,000, 5,000, and 10,000 observations for both SEMA and standard (offline) EM. For each simulation the population values were $\mu = 10$ and $\sigma^2 = 100$. The two factors that varied are τ^2 (presented in the columns) and n_j (presented in the rows). In bold are the parameter estimates that differed by more than 10 compared to the population values that generated the data.

Across all conditions, the SEMA and EM estimates of μ are close to the population value even with as little as 100 observations (see Table 2). However, the SEMA estimates are clearly much more variable than those of EM at the beginning of the data stream, that is, when SEMA did not have the chance to converge. But this difference has disappeared by 10,000 observations. Both EM and SEMA have larger SD's at the end of the data stream in the condition with $\tau^2 = 100$ and $n_j = 100$ than in the other conditions. This is due to the smaller number of individuals in this condition ($J = 100$), causing the data-generating model to fluctuate more

Table 1: Average reliability $\bar{\rho}$ in the simulation study for $\sigma^2 = 100$

	τ^2			
n_j	1	10	25	100
10	.091	.500	.714	.909
25	.200	.714	.862	.962
100	.500	.909	.962	.991

across the different runs.

The results for the estimated residual variance, σ^2, are presented in Table 3. In the condition in which τ^2 is low, both EM and SEMA underestimate the residual variance in the beginning of the data stream (when $n = 100$). SEMA somewhat overestimates σ^2 halfway through the data stream for the conditions where $\tau^2 > 1$. In all conditions, the variability of both the SEMA and EM estimate is large in the beginning of the data stream, but decreases steadily towards the end of the data stream. Irrespective of the average reliability, $\bar{\rho}$, at the end of the data stream ($n = 10,000$) σ^2 is estimated equally well by SEMA and EM.

Next, Table 4 presents the results for τ^2. In the five lowest reliability conditions ($\tau^2 = 1$ with $n_j = \{10, 25, 100\}$ and $\tau^2 = 10$ with $n_j = \{10, 25\}$, see Table 1), SEMA seems to slightly overestimate τ^2. However, when n = 5,000, SEMA starts approaching the EM estimates. The only situation in which τ^2 still seems overestimated at 10,000 observations occurs with $\tau^2 = 1$ and $n_j = 10$, which is the lowest reliability condition corresponding to $\bar{\rho} = 0.091$. In general, the higher the reliability the faster the estimate of τ^2 converges to its true value.

The last result we present from this simulation study is the average squared prediction error of the individual-level effects, $\bar{e^2}$. Table 5 shows that irrespective of the condition, $\bar{e^2}$ and its variability across replications are very large in beginning of the data stream for both SEMA and EM, but both the size and the variability of $\bar{e^2}$ decrease rapidly during the data stream; that is, when the model parameter estimates improve and the amount of information available per individual increases. The prediction quality of SEMA is similar to that of EM at 5,000 data points, except for the lowest reliability condition ($\bar{\rho} = 0.091$) in which SEMA performs somewhat worse. When the reliability increases, as expected, the prediction error decreases for both SEMA and EM. For a stream of length $n = 10,000$ the performance of EM and SEMA is identical.

4.2. Simulation study II: Improving SEMA in low reliability cases

4.2.1. Design

Our first simulation study showed that in the lowest reliability condition, i.e., when $n_j = 10$ and $\tau^2 = 1$, SEMA performs less well than EM. That is, the average estimates of SEMA for τ^2 are too high (at $n = 10,000$: SEMA: $\hat{\tau}^2 = 5.47$, EM: $\hat{\tau}^2 = 1.04$) and for σ^2 are too low (at $n = 10,000$: SEMA: $\hat{\sigma}^2 = 98.08$, EM: $\hat{\sigma}^2 = 100.00$) and moreover, the average squared prediction error, $\bar{e^2}$, of SEMA ($\bar{e^2} = 1.92$) is larger compared to EM ($\bar{e^2} = 0.94$).

One possible explanation for the fact that SEMA has some difficulties in the low reliability condition is that it is sensitive to the starting values, especially when the average reliability is very low. Our rather crude starting values may have been too

Table 2: estimates of μ (population value $\mu = 10$) averaged over 1,000 replications ($\sigma^2 = 100$). In the parentheses are the SD's over 1,000 replications.

		population values of τ^2															
		$\tau^2 = 1$				$\tau^2 = 10$				$\tau^2 = 25$				$\tau^2 = 100$			
		SEMA		EM		SEMA		EM		SEMA		EM		SEMA		EM	
$\bar{n}_j$	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
10	100	10.10	(3.90)	9.99	(1.05)	10.08	(4.13)	9.99	(1.09)	10.10	(4.46)	9.99	(1.16)	10.10	(5.92)	9.99	(1.46)
	1,000	10.07	(2.67)	10.00	(0.32)	10.06	(2.81)	10.00	(0.34)	10.08	(3.04)	10.00	(0.38)	10.12	(4.09)	10.00	(0.52)
	5,000	10.01	(0.89)	10.00	(0.15)	10.01	(0.92)	10.00	(0.17)	10.01	(0.97)	10.00	(0.21)	10.03	(1.07)	10.00	(0.34)
	10,000	10.00	(0.22)	10.00	(0.10)	10.00	(0.19)	10.00	(0.14)	10.00	(0.19)	10.00	(0.18)	10.00	(0.32)	10.00	(0.32)
25	100	9.82	(3.95)	10.06	(1.05)	9.84	(4.14)	10.07	(1.06)	9.85	(4.42)	10.07	(1.13)	9.91	(5.81)	10.10	(1.43)
	1,000	9.91	(2.01)	10.00	(0.32)	9.93	(2.11)	10.01	(0.37)	9.93	(2.22)	10.01	(0.43)	9.96	(2.98)	10.02	(0.63)
	5,000	10.00	(0.17)	10.00	(0.15)	10.00	(0.22)	10.01	(0.22)	10.01	(0.29)	10.01	(0.29)	10.01	(0.52)	10.01	(0.52)
	10,000	10.00	(0.12)	10.00	(0.12)	10.01	(0.19)	10.01	(0.19)	10.01	(0.27)	10.01	(0.27)	10.01	(0.51)	10.01	(0.51)
100	100	9.93	(3.56)	9.98	(1.03)	9.95	(3.73)	10.01	(1.12)	9.99	(4.03)	10.03	(1.25)	10.09	(5.23)	10.09	(1.70)
	1,000	10.01	(0.93)	10.01	(0.34)	10.04	(1.05)	10.03	(0.47)	10.08	(1.24)	10.05	(0.62)	10.15	(1.76)	10.10	(1.09)
	5,000	10.01	(0.18)	10.01	(0.18)	10.03	(0.35)	10.03	(0.35)	10.04	(0.53)	10.04	(0.53)	10.09	(1.04)	10.09	(1.04)
	10,000	10.01	(0.15)	10.01	(0.15)	10.03	(0.34)	10.03	(0.34)	10.05	(0.53)	10.05	(0.53)	10.10	(1.01)	10.10	(1.04)

Table 3: Estimates of σ^2 averaged over 1,000 replications, $(\mu = 10, \sigma^2 = 100)$. In the parentheses are the SD's over 1,000 replications and in bold those values which are more than 10 from the true value.

		population values of τ^2															
		$\tau^2 = 1$				$\tau^2 = 10$				$\tau^2 = 25$				$\tau^2 = 100$			
		SEMA		EM		SEMA		EM		SEMA		EM		SEMA		EM	
$\bar{n}_j$	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
10	100	98.23	(35.04)	**74.87**	(30.82)	107.40	(38.39)	**78.11**	(33.73)	**122.24**	(44.24)	**82.52**	(38.26)	**198.37**	(73.47)	95.12	(55.96)
	1,000	96.47	(20.26)	97.91	(5.31)	103.18	(22.55)	100.00	(6.60)	**114.29**	(26.49)	100.27	(7.13)	**170.38**	(49.62)	100.27	(7.46)
	5,000	96.61	(3.42)	99.85	(2.16)	99.60	(4.54)	100.08	(2.27)	103.01	(6.51)	100.08	(2.27)	109.48	(15.04)	100.08	(2.28)
	10,000	98.08	(1.48)	100.00	(1.48)	99.79	(1.56)	100.03	(1.50)	100.23	(1.57)	100.03	(1.50)	100.13	(1.51)	100.02	(1.50)
25	100	98.85	(33.41)	**85.61**	(21.79)	107.50	(36.36)	**88.58**	(24.74)	**121.39**	(40.72)	92.33	(28.93)	**193.94**	(72.38)	99.56	(40.14)
	1,000	96.33	(11.60)	98.75	(4.87)	101.03	(14.15)	99.89	(5.57)	107.98	(17.70)	99.90	(5.67)	**139.70**	(40.37)	99.87	(5.77)
	5,000	98.47	(2.03)	99.92	(2.08)	99.84	(2.10)	99.96	(2.10)	100.00	(2.11)	99.96	(2.10)	99.97	(2.10)	99.96	(2.10)
	10,000	99.51	(1.41)	99.93	(1.44)	99.94	(1.44)	99.94	(1.44)	99.94	(1.44)	99.94	(1.44)	99.94	(1.44)	99.94	(1.44)
100	100	98.40	(29.94)	92.77	(15.72)	105.21	(34.93)	95.48	(17.65)	**116.66**	(42.86)	97.68	(20.03)	**173.49**	(72.73)	99.00	(22.93)
	1,000	98.71	(16.66)	99.65	(4.62)	100.70	(20.88)	100.00	(4.80)	101.70	(25.59)	100.00	(4.80)	103.33	(42.81)	100.00	(4.81)
	5,000	99.95	(2.05)	99.99	(2.06)	100.00	(2.06)	100.00	(2.06)	100.00	(2.06)	100.00	(2.06)	100.00	(2.06)	100.00	(2.06)
	10,000	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)

Table 4: Estimates of τ^2 averaged over 1,000 replications, ($\mu = 10, \sigma^2 = 100$). In the parentheses are the SD's over 1,000 replications and in bold those values which are more than 10 from the true value.

		population values of τ^2															
		$\tau^2 = 1$				$\tau^2 = 10$				$\tau^2 = 25$				$\tau^2 = 100$			
		SEMA		EM		SEMA		EM		SEMA		EM		SEMA		EM	
$\bar{n}_j$	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
10	100	**18.76**	(11.55)	**26.11**	(28.80)	**20.33**	(12.51)	**31.75**	(32.27)	23.13	(14.46)	**42.12**	(37.63)	**36.24**	(24.22)	103.40	(59.83)
	1,000	**17.07**	(8.88)	3.40	(2.66)	18.99	(10.02)	10.20	(5.10)	22.35	(12.17)	24.86	(6.56)	**39.02**	(23.78)	99.62	(10.78)
	5,000	10.47	(3.22)	1.21	(0.68)	14.19	(4.41)	9.98	(1.37)	22.08	(6.90)	24.98	(2.08)	**78.85**	(20.20)	100.01	(5.53)
	10,000	5.47	(0.87)	1.04	(0.47)	11.00	(1.53)	10.01	(0.90)	24.32	(2.24)	25.02	(1.58)	99.42	(5.09)	100.05	(5.01)
25	100	**19.62**	(12.77)	**14.39**	(18.10)	**21.45**	(14.46)	**20.49**	(21.92)	24.42	(16.73)	31.67	(27.77)	**38.22**	(27.44)	98.68	(47.36)
	1,000	**15.13**	(6.87)	2.00	(1.83)	17.79	(8.32)	9.82	(3.79)	22.68	(10.81)	24.75	(5.13)	**51.44**	(27.11)	99.46	(11.28)
	5,000	4.20	(0.57)	1.00	(0.58)	10.40	(1.37)	9.92	(1.29)	24.68	(2.47)	24.85	(2.39)	99.42	(7.90)	99.51	(7.88)
	10,000	1.80	(0.22)	1.00	(0.35)	9.95	(1.03)	9.95	(1.03)	24.88	(2.14)	24.88	(2.14)	99.51	(7.64)	99.51	(7.64)
100	100	**17.61**	(11.34)	7.17	(9.47)	19.50	(12.68)	13.23	(13.42)	22.96	(15.19)	25.70	(18.85)	**40.22**	(28.04)	98.00	(34.68)
	1,000	5.43	(1.35)	1.20	(1.23)	10.80	(2.95)	9.72	(2.93)	23.61	(5.85)	24.54	(5.05)	96.73	(18.83)	98.67	(15.63)
	5,000	1.05	(0.37)	0.98	(0.42)	9.88	(1.72)	9.87	(1.72)	24.71	(3.86)	24.71	(3.86)	98.86	(14.52)	98.85	(14.52)
	10,000	0.98	(0.29)	0.98	(0.29)	9.88	(1.57)	9.88	(1.57)	24.71	(3.70)	24.71	(3.70)	98.86	(14.34)	98.86	(14.34)

Table 5: The average squared error ($\bar{e^2} = \sum_{j=1}^{J} (\hat{\mu}_j - \mu_j)^2/J$) averaged over 1,000 replications. In the parentheses are the SD's over 1,000 replications.

		population values of τ^2															
		$\tau^2 = 1$				$\tau^2 = 10$				$\tau^2 = 25$				$\tau^2 = 100$			
		SEMA		EM		SEMA		EM		SEMA		EM		SEMA		EM	
$\bar{n}_j$	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
10	100	20.43	(27.19)	16.43	(23.88)	28.06	(29.93)	23.44	(22.19)	40.80	(34.65)	33.62	(19.48)	106.75	(63.51)	65.08	(16.99)
	1,000	11.73	(16.96)	1.26	(0.40)	18.26	(18.89)	9.05	(0.67)	29.18	(22.12)	18.66	(1.23)	82.87	(43.91)	42.16	(2.62)
	5,000	4.16	(2.62)	0.99	(0.06)	8.35	(3.45)	6.84	(0.32)	13.70	(5.02)	11.80	(0.57)	26.10	(13.32)	19.33	(0.98)
	10,000	1.92	(0.37)	0.94	(0.05)	5.24	(0.29)	5.15	(0.24)	7.63	(0.40)	7.56	(0.35)	10.08	(0.53)	10.01	(0.48)
25	100	21.12	(26.19)	7.16	(11.35)	28.49	(28.55)	15.14	(10.81)	40.55	(32.72)	26.30	(10.09)	103.92	(63.21)	56.48	(13.67)
	1,000	8.58	(9.76)	1.17	(0.28)	13.96	(11.79)	8.20	(0.68)	22.06	(14.71)	15.69	(1.20)	57.81	(36.65)	31.05	(2.36)
	5,000	1.53	(0.21)	0.93	(0.08)	4.59	(0.34)	4.57	(0.33)	6.36	(0.47)	6.35	(0.47)	7.98	(0.60)	7.98	(0.60)
	10,000	0.89	(0.08)	0.82	(0.06)	2.92	(0.21)	2.92	(0.21)	3.56	(0.26)	3.56	(0.26)	4.00	(0.29)	4.00	(0.29)
100	100	17.89	(26.17)	3.57	(3.79)	24.63	(30.97)	11.46	(3.99)	36.01	(38.36)	21.70	(5.02)	90.48	(66.36)	45.24	(9.68)
	1,000	2.74	(15.29)	1.10	(0.27)	6.29	(19.35)	5.30	(0.80)	9.13	(23.92)	7.63	(1.13)	13.17	(40.94)	9.98	(1.50)
	5,000	0.70	(0.11)	0.71	(0.11)	1.70	(0.25)	1.70	(0.25)	1.89	(0.27)	1.89	(0.27)	2.01	(0.29)	2.01	(0.29)
	10,000	0.52	(0.08)	0.52	(0.08)	0.92	(0.13)	0.92	(0.13)	0.98	(0.14)	0.98	(0.14)	1.01	(0.15)	1.01	(0.15)

far off to guarantee convergence within 10,000 data points. A second explanation is that this low reliability condition is a rather difficult condition even for EM. That is, in a situation where the average reliability equals .091, the EM algorithm needs hundreds of iterations (and passes through the full dataset) to converge. It is not surprising that the SEMA algorithm, which passes through the dataset only once, has not yet reached the peak of the likelihood function.

These two explanations suggest two possible adaptations of SEMA: a) an adaptation yielding better starting values and b) an adaptation in which more than one pass over all individuals is performed. For this purpose, we investigate three possible variants of the SEMA algorithm, which we refer to as SEMA-T, SEMA-U, and SEMA-TU, where the T refers to *training*, and the U to *update*. That is,

1. **SEMA-T**: While SEMA is used to obtain estimates of the individual-level effect and the model parameters, when the *first* 1,000 observations of the data stream have entered, the EM algorithm (which iterates until convergence) is used to obtain better estimates for the model parameters, which are subsequently used for SEMA.

2. **SEMA-U**: A single EM iteration over all available individuals is used to update all the estimated individual-level effects and model parameters after *each* 1,000 data points.

3. **SEMA-TU**: combines both features.

The training set could provide SEMA with better starting values, speeding up the convergence to a local maximum. The second variant of SEMA, using EM updates is especially useful when observations of an individual enter in a block. In that case the contributions to the CDSS will be based on model parameters which are not yet converged, and more importantly these erroneous contributions to the CDSS are not corrected, because this individual is no longer returning. Doing an additional full E step will help in correcting the contributions to the CDSS. In this second simulation study, we repeat the $n_j = 10$ and $\tau^2 = 1$ condition but now we also apply these three variants of SEMA. Additionally we keep track of the computational time required by each of the different algorithms: EM, SEMA, and the three variants of SEMA.

4.2.2. Results

Table 6 presents the results obtained with the different variants of SEMA at $n =$ 900, 1,000, 5,000, and 10,000 observations. At 900 observations, all SEMA versions are still identical, but at 1,000 observations large differences appear between the variants using those observations as a training set and those that do not. For μ,

the average estimates were already close to the true value at $n = 900$ observations, but clear improvements are visible in the SDs, with the variants of SEMA with a training set having lower SDs than those without a training set. At 5,000 observations, the difference between EM and SEMA-T and SEMA-TU are minimal. The training set in SEMA-T and SEMA-TU clearly improves the precision of the estimates of μ, while the additional update only marginally improves the precision.

However, for τ^2, the SEMA variants have a large impact on both the point estimate and their SD. Allowing for a single EM updates every 1,000 data points (SEMA-U) already yields a solution that is much closer to the full EM solution. An even larger improvement is shown by SEMA-T. Using both a training set and EM updates yields another slight improvement of the estimate of τ^2. A similar pattern can be observed for the residual variance σ^2, though the effect is smaller because the SEMA estimate was already close to the true value. Using a training set and EM updates yields estimates closer to those of the EM algorithm, though the additional updates seem to only have a minimal influence on the estimate and its SD.

The average squared prediction error is more affected by using a training set or additional EM updates. This effect of the training set and updates is especially noticeable halfway through the data stream. The variant with only the training dataset outperforms the variant with only the updates. Towards the end of the data stream, the difference between standard SEMA and its variants becomes much smaller.

Finally, Figure 1 presents the difference in cumulative computation time when the algorithms have to produce up-to-date parameter estimates each time a new data point arrives (or after the indicated number of data points). We scale the time required to update the model parameters proportional to the time required to estimate the model when $n = 500$. There is no visible difference between the different variants of SEMA, which all grow linearly by factor of about 10 (as n grows with a factor of 20). Figure 1 shows four variants of the EM algorithm. The model parameters are updated using EM every: 1, 10, 100, or 1,000 data points. All four variants of the EM algorithm grow with a much larger factor than SEMA when it has to produce up-to-date parameter estimates when data enter over time. More importantly the curves of the EM algorithm tend to deviate from linear and curve more upwards as larger datasets are analyzed. These curved lines of the EM algorithm illustrate that analyzing nested data using the EM algorithm when data points enter over time becomes infeasible, as the estimation of the model parameters will require increasingly more time.

To conclude, both the model-parameter estimates and the prediction errors can be improved by using better starting values obtained from a training dataset. Also, performing a single EM iteration after every 1,000 data points improved parameter

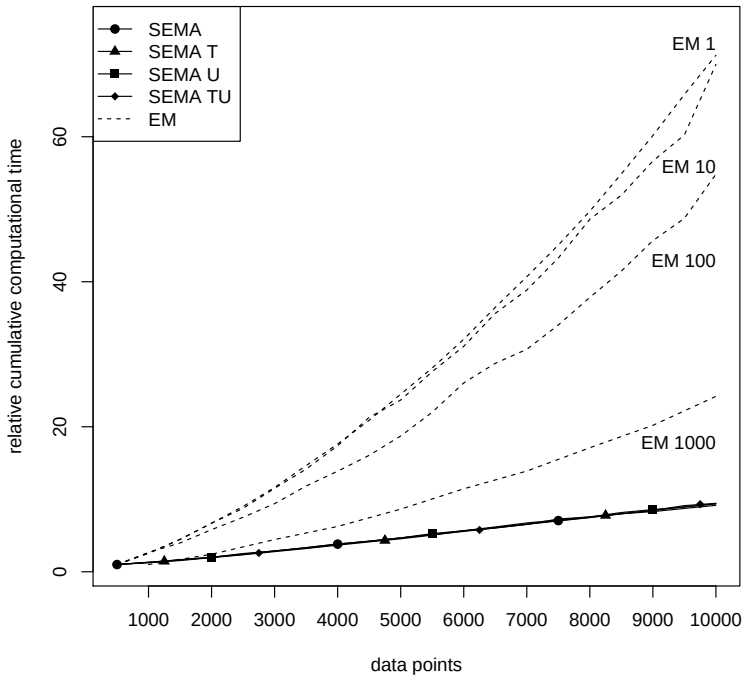

Figure 1: Relative cumulative computation time as a function of increasing number of data points, when parameters are updated after each data point (SEMA, EM 1, or after the indicated number of data points (10, 100, or 1,000).

Table 6: Results of SEMA variants in the condition $\mu = 10$, $\tau^2 = 1$, and $\sigma^2 = 100$. In the parentheses are the SD's over 1,000 replications, and in bold those values which are more than 10 from the population value: $\tau^2 = 1$.

		SEMA		SEMA T		SEMA U		SEMA T+U		EM	
	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
$\hat{\mu}$	900	10.07	(2.74)	10.07	(2.74)	10.07	(2.74)	10.07	(2.74)	10.00	(0.33)
	1,000	10.07	(2.67)	10.00	(0.32)	10.06	(2.32)	10.00	(0.32)	10.00	(0.32)
	5,000	10.01	(0.89)	10.00	(0.24)	10.00	(0.41)	10.00	(0.21)	10.00	(0.15)
	10,000	10.00	(0.22)	10.00	(0.16)	10.00	(0.13)	10.00	(0.12)	10.00	(0.10)
$\hat{\tau}^2$	900	**17.24**	(9.09)	**17.24**	(9.09)	**17.24**	(9.09)	**17.24**	(9.09)	3.64	(2.89)
	1,000	**17.07**	(8.88)	3.42	(2.71)	**16.18**	(7.92)	3.42	(2.71)	3.40	(2.66)
	5,000	10.47	(3.22)	3.10	(2.10)	7.74	(1.72)	2.92	(1.80)	1.21	(0.68)
	10,000	5.47	(0.87)	2.50	(1.25)	3 .67	(0.41)	2.16	(0.87)	1.04	(0.47)
$\hat{\sigma}^2$	900	96.47	(21.03)	96.47	(21.03)	96.47	(21.03)	96.47	(21.03)	97.61	(5.72)
	1,000	96.47	(20.26)	97.77	(5.33)	95.17	(17.60)	97.77	(5.33)	97.91	(5.31)
	5,000	96.61	(3.42)	98.64	(2.36)	96.45	(2.32)	98.63	(2.31)	99.85	(2.16)
	10,000	98.08	(1.48)	99.16	(1.54)	98.42	(1.45)	99.19	(1.50)	100.00	(1.48)
$\bar{e^2}$	900	12.12	(17.52)	12.12	(17.52)	12.12	(17.52)	12.12	(17.52)	1.30	(0.46)
	1,000	11.73	(16.96)	1.27	(0.41)	9.42	(14.05)	1.27	(0.41)	1.26	(0.40)
	5,000	4.16	(2.62)	1.28	(0.41)	2.45	(0.69)	1.22	(0.31)	0.99	(0.06)
	10,000	1.92	(0.37)	1.14	(0.24)	1.33	(0.17)	1.05	(0.14)	0.94	(0.05)

estimates and lowered prediction errors. Experimentation with variants of the latter method showed that even larger improvements can be obtained by either performing multiple EM iterations, or performing the single EM iteration more frequently. In other words, depending on whether this is feasible in the streaming data application concerned, other combinations of SEMA and EM could be used.

5. An application of SEMA to longitudinal happiness ratings

To illustrate the use of SEMA in a real-life application, we use data from a longitudinal study of happiness ratings by Killingsworth and Gilbert (2010). We will fit a random-intercept model to the data to obtain individual-level estimates of respondents' happiness. Data were collected using a smart-phone application, yielding daily measurements of the participants' happiness on a continuous rating scale. The dataset contains a total of $n = 17{,}742$ observations for $J = 2{,}248$ individuals. The average number of observation per person is 7.89, with a minimum of one observation (254 individuals) and maximum of 39 observations (one individual). While the authors analyzed the dataset after the data collection stopped, in reality, the data entered as a data stream. We fit a random-intercept model on the data stream resulting from the smart-phone application, by replaying the data collection over time.

Table 7: Longitudinal happiness ratings: model parameter estimates and average squared error

	$\hat{\mu}$		$\hat{\tau}^2$		$\hat{\sigma}^2$		$\bar{e^2}$	
n	SEMA	EM	SEMA	EM	SEMA	EM	SEMA	EM
100	68.15	67.01	125.60	161.52	235.87	229.08	31.40	27.15
1,000	65.60	65.49	129.24	121.72	349.30	353.06	24.64	22.82
5,000	63.96	63.80	103.33	93.91	336.33	337.67	19.58	19.06
10,000	64.34	64.39	103.61	97.16	357.36	359.90	14.14	13.68
17,742	64.72	64.85	100.22	93.58	365.28	367.16	0.30	–

During the data stream, we obtained parameter estimates from the SEMA and EM algorithms from only the data seen so far, and compared these to the EM estimates using the entire dataset: $\hat{\mu} = 64.85$, $\hat{\tau}^2 = 93.58$, and $\hat{\sigma}^2 = 367.16$. We used the individual-level effects estimated using all data (i.e., end of the data stream) as the "true" individual-level effect, to compute the average squared prediction error during the data stream. From these "true" estimates we find that the estimated reliabilities range from 0.20 to 0.91 with an average of 0.67. As in the simulation study, during the data stream we monitored the estimates of $\hat{\mu}$, $\hat{\tau}^2$, and $\hat{\sigma}^2$, as well as the estimated individual-level effects. The starting values for SEMA were, respectively, $\mu_0 = y_{t=1}$ (the first observation), $\tau_0^2 = 1$, and $\sigma_0^2 = 1$.

Table 7 reports the values of the parameter estimates and the average squared prediction error for both SEMA and EM. Similar to the results obtained in the simulation study, $\hat{\mu}$ is estimated properly, while $\hat{\tau}^2$ is somewhat overestimated by SEMA compared to EM. The residual variance, $\hat{\sigma}^2$, is somewhat underestimated. The average squared prediction error of SEMA is close to the average squared prediction error of the EM algorithm, even though at the end of the data stream EM is obviously favored due to its own use in the operationalization of the "gold standard".

6. SEMA characteristics

6.1. Theoretical considerations

The proposed SEMA algorithm yields two improvements compared to the traditional EM algorithm. First, it is no longer required to store all n data points in memory, leading to a decrease in memory required. What needs to be stored are merely the current values of n_j, $\bar{y}_j$, $\bar{y_j^2}$, T_{1j}, T_{2j}, and T_{3j} for each of the J individuals. Second, SEMA decreases the number of computations compared to the conventional EM algorithm when analyzing a data stream. The SEMA algorithm updates the n_{j_t}, $\bar{y}_{j_t}$, $\bar{y^2_{j_t}}$, T_{1j_t}, T_{2j_t}, and T_{3j_t} for a single individual only, and subsequently updates the CDSS and the model parameters in a single pass.

SEMA is conceptually positioned between what Liang and Klein (2009) call incremental EM and stepwise EM (Cappé and Moulines, 2009). Neal and Hinton (1998) provide a proof for the large sample convergence of both stepwise and incremental EM. Incremental EM estimates the parameters by storing the CDSS and the contributions of each of the *data points* and then iterates over the dataset, subtracting previous contributions of the data point to the CDSS, thereby correcting the erroneous contribution to the CDSS of previous time. As such, incremental EM requires all data points in memory. In contrast, stepwise EM does not store all the data, but it does not correct for previous contributions to the CDSS: stepwise EM adds a weighted contribution of each data point to the CDSS. To use stepwise EM the analyst has to choose a weight given to new data points. SEMA, which conceptually combines the two earlier methods, does not store the observed data; it stores only contributions at the level of the individuals, instead of the data points themselves. This means that SEMA scales with J, instead of n in the case of incremental EM, while using more information than stepwise EM.

6.2. Convergence

Fitting multilevel models on data streams adds an additional complication to standard offline methods: it is not immediately clear when (e.g., after how many observations), the parameter estimates can be said to have "converged" and thus can be substantively interpreted. However, options are available to address this issue. One could, for example choose, during the data stream, to compute a moving average of the absolute difference between two estimates of the same parameter at adjacent time points:

$$\bar{\delta}_\theta = \sum_{i=(t-C)}^{(t-1)} |\hat{\theta}_i - \hat{\theta}_{i+1}|/C, \tag{18}$$

where C is the size of the window of the moving average and θ is one of the model parameters. As new data points enter and thus t increases, the average will cover a new interval of parameter differences. This measure $\bar{\delta}_\theta$ – which can be maintained during the stream – can be used to quantify convergence (where given some cut-off ζ, $\bar{\delta}_\theta < \zeta$ would imply convergence). If we examine the behavior of $\bar{\delta}_\theta$ for the simulations presented in Section 4, we find for the parameter μ that in all streams $\bar{\delta}_\mu$ monotonically approaches zero, and that the difference between the parameter estimates of μ obtained using our online method, and those obtained offline (where we determined convergence by no change in parameter estimates to the fourth decimal) decreases as $\bar{\delta}_\mu$ decreases. Hence, $\bar{\delta}_\theta$ seems a good candidate to use for convergence; the smaller the value of $\bar{\delta}_\theta$, the closer the parameter estimates are to their offline equivalent. The actual cut-offs ζ for $\bar{\delta}_\theta$ will be problem dependent and might differ for the different parameters. For the parameter σ^2 we also find that

$\bar{\delta}_{\sigma^2}$ decreases during the stream, and that this decrease corresponds to more and more precise estimates. However, for σ^2, the decrease is quite a bit slower (i.e., σ^2 needs more observations) than for μ, in many of the simulations, indicating that some parameters might be said to have converged sooner than others.

7. Extending SEMA

In practice one might want to extend the random-intercept model to a model with more parameters. One could, for example, include covariates to improve the estimates of the model parameters and the predictions resulting from the model. Here we discuss the inclusion of additional *fixed* effects to SEMA. This model can be written as:

$$y_{ij} = \mathbf{x}_j\beta + \mu_j + \epsilon_{ij}, \tag{19}$$

where $\mathbf{x}_j$ is a p-dimensional row-vector of covariates at the individual level with first element equal to 1 for the intercept, β is a p-dimensional vector with fixed-effect regression coefficients, and the individual-level intercepts μ_j are normally distributed as:

$$\mu_j \sim \mathcal{N}(0, \tau^2). \tag{20}$$

We assume the covariates are constant within each individual: $x_{ij} = x_j$. Because μ_j is now centered around zero, the computation of the parameters is altered slightly. In the E step the following individual-level parameters are computed:

$$\hat{\mu}_j = \hat{V}_j^{-1}(\bar{y}_j - \mathbf{x}_j\hat{\beta})n_j, \tag{21}$$

where $\hat{V}_j^{-1}$ equals

$$\begin{aligned} \hat{V}_j^{-1} &= \hat{\nu}_j/\hat{\sigma}^2, \\ &= \hat{\tau}^2\Big(1 - \frac{\hat{\tau}^2}{\hat{\tau}^2 + {\hat{\sigma}^2}/{n_j}}\Big)\frac{1}{\hat{\sigma}^2}, \\ &= \frac{1}{{\hat{\sigma}^2}/{\hat{\tau}^2} + n_j}. \end{aligned} \tag{22}$$

The CDSS of β, τ^2, and σ^2 are again referred to as T_1, T_2, and T_3, where T_1 is now a vector, instead of a scalar. The contributions to the CDSS for a single individual are can then be computed as follows:

$$T_{1j_t} = n_{j_t}\mathbf{x}'_{j_t}\hat{\mu}_{j_t}, \tag{23}$$

$$T_{2j_t} = \hat{\mu}^2_{j_t} + \hat{\nu}_{j_t}, \tag{24}$$

$$T_{3j_t} = [(\bar{y}_{j_t} - \mathbf{x}_{j_t}\hat{\beta} - \hat{\mu}_{j_t})^2 + \hat{\nu}_{j_t}]n_{j_t}, \tag{25}$$

where, as previously, $T_{w(t)} = T_{w(t-1)} - T_{wj_t(t-1)} + T_{wj_t(t)}$.

In the M step, the CDSS can be used to obtain new estimates for the model parameters using Escobar and Moser's (1993) updating method for the regression coefficients, as follows:

$$\begin{aligned}
\hat{\beta} =& \Big(\sum_{j=1}^{J} \sum_{i=1}^{n_j} (\mathbf{x}'_{ij} \mathbf{x}_{ij}) \Big)^{-1} \sum_{j=1}^{J} \sum_{i=1}^{n_j} \mathbf{x}'_{ij} (y_{ij} - \hat{\mu}_j), \\
=& \Big(\sum_{j=1}^{J} n_j (\mathbf{x}'_j \mathbf{x}_j) \Big)^{-1} \sum_{j=1}^{J} \mathbf{x}'_j (\bar{y}_j - \hat{\mu}_j) n_j, \\
=& \Big(\sum_{j=1}^{J} n_j (\mathbf{x}'_j \mathbf{x}_j) \Big)^{-1} \sum_{j=1}^{J} n_j \mathbf{x}'_j \bar{y}_j - n_j \mathbf{x}'_j \hat{\mu}_j, \\
=& A_1 (A_2 - T_1),
\end{aligned} \tag{26}$$

where A_1 and A_2 can both be computed online:

$$A_{1(t)} = A_{1(t-1)} - \frac{A_{1(t-1)} \mathbf{x}_{ij_t} \mathbf{x}'_{ij_t} A_{1(t-1)}}{1 + \mathbf{x}'_{ij_t} A_{1(t-1)} \mathbf{x}_{ij_t}}, \tag{27}$$

$$A_{2(t)} = A_{2(t-1)} + \mathbf{x}'_{ij_t} y_{ij_t}. \tag{28}$$

Note that we use $\mathbf{x}_{ij_t}$ in this formulation. This means that every time a new data point arrives the values of the covariates $\mathbf{x}_{j_t}$ are retrieved from memory. Because A_1 and A_2 only consist of observed data (there are no model parameters involved) and it are sums over n observations, we do not have to correct for previous contributions.

Moreover, using the notation including the summation over n_j, the fixed effect is weighted according to the number of observations of an individual. Taking into account which individuals have more observations results in better estimates of β in the case where the individual-level effect μ_j is dependent on the number of observations of that individual, n_j. For the model introduced in Equation 3, if μ_j depends on n_j, $T_{1j_t} = n_j \hat{\mu}_j$ and $\hat{\mu} = T_1/n$. In our simulation studies the individual-level effects were not dependent on the number of observations. Therefore the results using either of the two formulations will effectively be the same.

Next, the variance of the random effect, $\hat{\tau}^2$, is computed as follows:

$$\hat{\tau}^2_{(t)} = \frac{T_{2(t)}}{J}. \tag{29}$$

This is slightly different from the previous formulation in Equation 12; the difference is due to the fact that μ_j is now distributed around 0 instead of μ, since we

separated the fixed effects from the random effects. Lastly, the residual variance $\hat{\sigma}^2$ is the same as it was previously,

$$\hat{\sigma}^2_{(t)} = \frac{T_{3(t)}}{J}. \tag{30}$$

Other interesting extensions concern the inclusion of fixed and random effects for level-1 predictors and the generalization to more than 2 levels of nesting. For those models, SEMA versions can also be formulated, which as shown here involves the derivation of the updating formulas for the expected sufficient statistics and for the parameters. In future research we will look into these extensions.

8. Discussion

Since data streams are becoming more common in both real-life applications and social science research (e.g., Hofmann, Adriaanse, Vohs, and Baumeister, 2014; Killingsworth and Gilbert 2010; San Pedro, Baker, Bowers, and Heffernan, 2013) there is a need for computationally feasible methods to analyze data streams. This paper presents a novel method for estimating multilevel models in data streams consisting of dependent observations. Because the regular EM algorithm becomes computationally infeasible as the size of the data stream grows, we propose a streaming EM approximation (SEMA). SEMA is obtained by adapting the E step of the EM algorithm; that is, by using a partial E step (McLachlan and Peel, 2000) in which only the contributions to the sufficient statistics of the individual providing the new observation are updated.

Our first simulation study showed that SEMA recovers both the fixed effect and the individual-level (random) effects well, as encountered in grouped data streams. Also, the variance components are well estimated, although in conditions with very low reliability (i.e., when the residual variance is large compared to the variance of the random intercept), a large number of data points are needed to get estimates which are close to population values. In the second simulation study, we examined two ways to improve the estimates obtained by SEMA early in the data stream. First, one could occasionally preform a single EM iteration using all individuals entered so far. Using this extra information for the estimation of the model parameters resulted in parameters which approached the EM estimates of the parameters faster. Second, one could use the first n (where we choose $n = 1,000$) data points of the stream as a training set. These first data points can be used to obtain better starting values, by applying EM until convergence, after which the stream is continued using SEMA to estimate the model parameters. The combination of the two approaches showed an even larger improvement. Finally, in our implementation, an individual-level effect is updated when a new data point

enters for the person concerned. However, when individual-level prediction is the main focus of the analysis, one could fine tune the estimation of the individual-level effects, for example, by recomputing these at the moment that they are needed using the most recent model parameter estimates. The proposed alterations to SEMA, SEMA-T and SEMA-U, provide a step in this direction.

It is to be noted that the random-intercept model, as presented in Equation 3, which provided the basis for our SEMA algorithm, can also be formulated differently: one could also interpret the current model as a factor analysis model in which our "observations within individuals" correspond to multiple items within individuals, to which one fits a single-factor model. The current model could then be specified as: $\mu + \tau z_j + \epsilon_{ij}$, where $z_j \sim \mathcal{N}(0, 1)$. The (offline) EM algorithm to fit this model, and its generalizations, is specified in detail in Rubin and Thayer (1982). For our current model the covariances between the items are constrained, which allows one to also in this formulation derive a online version of the EM algorithm leading to the same update steps as presented here. However, this seems not to be true in the general case: when the covariances are unconstrained, the computation of the sufficient statistics in a data stream seems cumbersome, due to the differing numbers of observations within individuals during the stream. Still, the factor-analytic view on the current problem might, in future work, inspire online EM approximations of more complex models.

Another issue to be noted is that ordering of the data points in the data stream is important for the rate of convergence of SEMA. Especially in the beginning of the data stream, if the data points are very extreme, SEMA will require more data to find the maximum-likelihood estimates for the model parameters. This is conceptually similar to using offline EM with poorly-chosen starting values of the parameters: in this case also convergence will be slow. As the data stream progresses, the influence of extreme values will lessen, since their contribution to the CDSS will decrease at a rate of at least $1/J$. Additionally, in the case that all the data for an individual enter as a block (i.e., all at once), the individual-level effect for this individual could be based on model parameters which are not yet close to the maximum of the likelihood function. This could result in contributions to the CDSS of an individual which are incorrect, and because the data entered in a block, the incorrect contributions to these CDSS are not corrected. Even though the effect of these incorrect contributions will decrease eventually, as new data points (and individuals) enter, this is an additional reason to do a full EM iteration, using all individuals, occasionally during the data stream.

With the introduction of SEMA, we provide a novel method to fit multilevel models row-by-row. This allows for the analysis of data streams and extremely large data sets, without revisiting the previous data. Because SEMA is an online method, it is not necessary to store all the data points in memory. Additionally,

SEMA requires less computational power than the EM algorithm when fitting multilevel models to data streams. These two advantages make SEMA attractive both in terms of the number of computations and in terms of the memory requirements.

Acknowledgements

We would like to thank Matthew A. Killingsworth and Daniel T. Gilbert for sharing their data. Furthermore we would like to thank the editor and the anonymous reviewers for the great contribution to the paper. Finally, we would like to thank James Mason, Sophia Rabe-Hesketh, and Anders Skrondal for their feedback during the writing process.

References

Barrett, P., Zhang, Y., Moffat, J., Kobbacy, K., 2013. A Holistic, Multi-level Analysis identifying the Impact of Classroom Design on Pupils' Learning. Building and Environment 59, 678–689.

Berlinet, A. F., Roland, C., 2012. Acceleration of the EM algorithm: P-EM versus epsilon algorithm. Computational Statistics & Data Analysis 56 (12), 4122–4137.

Browne, W., Goldstein, H., 2010. MCMC sampling for a multilevel model with nonindependent residuals within and between cluster units. Journal of Educational and Behavioral Statistics 35 (4), 453–473.

Cappé, O., Moulines, E., 2009. Online expectation-maximization algorithm for latent data models. Journal of the Royal Statistics Society: Series B (Statistical Methodology) 71 (3), 593–613.

Cortes, C., Fisher, K., Pregibon, D., Rogers, A., Smith, F., 2000. Hancock : A Language for Extracting Signatures from Data Streams. In: Proc. of the Sixth ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD). Boston, pp. 9–17.

Dempster, A. P., Laird, N. M., Rubin, D. B., 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological) 39 (1), 1–38.

Escobar, L., Moser, E., 1993. A Note on the Updating of Regression Estimates. The American Statistician 47 (3), 192–194.

Gaber, M. M., Zaslavsky, A., Krishnaswamy, S., 2005. Mining Data Streams : A Review. SIGMOD 34 (2), 18–26.

Gelman, A., 2007. Rich State, Poor State, Red State, Blue State: What's the Matter with Connecticut? Quarterly Journal of Political Science 2 (4), 345–367.

Hofmann, W., Adriaanse, M., Vohs, K. D., Baumeister, R. F., 2014. Dieting and the self-control of eating in everyday environments: An experience sampling study. British Journal of Health Psychology 19 (3), 523–539.

Killingsworth, M. A., Gilbert, D. T., 2010. A Wandering Mind Is an Unhappy Mind. Science 330 (6006), 932.

Lee, J., Podlaseck, M., Schonberg, E., Hoch, R., 2001. Visualization and Analysis of Clickstream Data of Online Stores for Understanding Web Merchandising. Data Mining and Knowledge Discovery 5 (2), 59–84.

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 on NAACL 09. p. 611.

Liu, Z., Almhana, J., Choulakian, V., McGorman, R., 2006. Online EM algorithm for mixture with application to internet traffic modeling. Computational Statistics & Data Analysis 50 (4), 1052–1071.

McLachlan, G., Peel, D., 2000. Finite Mixture Models. Wiley, New York.

Morris, C. N., Lysy, M., 2012. Shrinkage Estimation in Multilevel Normal Models. Statistical Science 27 (1), 115–134.

Neal, R., Hinton, G. E., 1998. A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants. In: Jordan, M. I. (Ed.), Learning in Graphical Models. pp. 355–368.

Ng, S. K., McLachlan, G. J., 2003. On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures. Statistics and Computing 13 (1), 45–55.

Opper, M., 1998. A Bayesian Approach to Online Learning. In: Saad, D. (Ed.), On-Line Learning in Neural Networks. Cambridge University Press, Cambridge, pp. 363–378.

Patidar, R., Sharma, L., 2011. Credit Card Fraud Detection Using Neural Network. International Journal of Soft Computing and Engineering 1, 32–38.

Plackett, R., 1950. Some Theorems in Least Squares. Biometrika 37, 149–157.

R Core Team, 2013. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Raudenbush, S., Bryk, A., 2002. Hierarchical Linear Models: applications and Data Analysis Methods, 2nd Edition. Sage Pulication, Thousand Oaks, California, USA.

Rubin, D. B., Thayer, D. T., 1982. Em algorithms for ml factor analysis. Psychometrika 47 (1), 69–76.

San Pedro, M. O. Z., Baker, R., Bowers, A. J., Heffernan, N. T., 2013. Predicting college enrollment from student interaction with an intelligent tutoring system in middle school. Proceedings of the 6th International Conference on Educational Data Mining, 177–184.

Steenbergen, M. R., Jones, B. S., 2002. Modeling Multilevel Data Structures. American Journal of Political Science 46 (1), 218–237.

Stein, C., 1956. Inadmissibility of the Usual Estimator for the Mean of a Multi-Variate Normal Distribution. In: Proc. Third Berkeley Symp. on Math. Statist. and Prob. Vol. 1. pp. 197–206.

Steiner, P. M., Hudec, M., 2007. Classification of large data sets with mixture models via sufficient EM. Computational Statistics and Data Analysis 51 (11), 5416–5428.

Thiesson, B., Meek, C., Heckerman, D., 2001. Accelerating EM for large databases. Machine Learning 45 (3), 279–299.

Welford, B., 1962. Note on a Method for Calculating Corrected Sums of Squares and Products. Technometrics 4 (3), 419–420.

Wolfe, J., Haghighi, A., Klein, D., 2008. Fully distributed EM for very large datasets. Proceedings of the 25th international conference on Machine learning - ICML '08, 1184–1191.