doi.org/10.26398/IJAS.0029-011

A FRAMEWORK FOR THE INCREMENTAL UPDATE OF THE MCA SOLUTION

Angelos Markos[1]

Department of Primary Education, Democritus University of Thrace, Greece

Alfonso Iodice D'Enza

Department of Economics and Law, University of Cassino and Southern Lazio, Italy

Abstract: *In the era of data deluge, a major challenge is to handle large amounts of data which are produced at a high rate and are characterized by association structures changing over time. As modern applications become increasingly scalable, efficient approaches are needed for dealing with high-dimensional categorical data. Multiple Correspondence Analysis (MCA) is a popular method for reducing the dimensionality of categorical data while preserving the most essential information. MCA is typically implemented via the eigenvalue decomposition (EVD) or the singular value decomposition (SVD) of a suitably transformed matrix. Because of the high computational and memory requirements of ordinary EVD and SVD, MCA is essentially unfeasible with massive data sets or data streams that change rapidly and have to be processed on the fly. We distinguish two main families of methods that can be efficiently used to incrementally compute the dominant eigenvalues and eigenvectors of a covariance matrix, i) stochastic approximation and ii) heuristic incremental EVD/SVD. A general algorithmic framework is presented to embed these methods in the MCA context and provide incremental dimension reduction of categorical data. The methods are compared on artificial data, in terms of the similarity between ordinary and incremental MCA configurations. Results do not clearly support the superiority of one method over another. However, methods that allow for block-based updates outperform vector-based approaches. The method of choice may be decided on the basis of the most desirable balance of speed and accuracy.*

Keywords: *Categorical data streams, Dimensionality reduction, Stochastic approximation, Incremental SVD.*

1. INTRODUCTION

Multiple Correspondence Analysis (MCA) is a well-established method for investigating the relationships between more than two categorical variables. Similar to Principal Component Analysis (PCA), MCA aims to identify a reduced set of

[1] Corresponding author: Angelos Markos, email: amarkos@eled.duth.gr

synthetic dimensions maximizing the explained variability of the categorical data set in question, see e.g. Greenacre (2007). More specifically, MCA leads to obtain a simplified low-dimensional representation of variable associations among groups of categorical variables. The first two or three principal components resulting from MCA are usually displayed as axes of a two or three-dimensional scatterplot, representing units and/or variable categories as points. The classic application of MCA is on the analysis of survey data, and it has been profitably used in fields ranging from marketing to psychology, to social and environmental sciences.

Recently new areas of application emerged, that usually involve large/massive amounts of categorical data. Some of the examples that can be given in this context are the continuous monitoring of typical product purchase combinations in market basket data, visualization of web-page visiting patterns via web-log analysis, tracking patient symptoms and behaviors over time, and monitoring of word associations that are present in data pulled on-the-fly from social networking sites. In all of these examples there is a high rate of data accumulation coupled with constant changes in data characteristics. Such type of data is often referred to as data streams or data flows. Data streams are usually large in volume, unbounded in size, dynamically changing and require fast response time and efficient memory use. In fact, when it comes to analyze very large data sets, or even data streams, the applicability of ordinary MCA is limited and requires a different approach. Such limitations depend on the eigenvalue decomposition (EVD) and singular value decomposition (SVD), as the MCA solution is obtained by performing one of such decompositions to a suitably transformed data matrix. In particular, the application of ordinary EVD and SVD to large and high-dimensional data is infeasible because of the high computational and memory requirements; in addition EVD/SVD, and hence MCA, are unsuitable for data flows, i.e., when new data arrive, one needs to re-run the method with the original data augmented by the new data and the whole data structures being decomposed have to be kept in memory.

A popular approach to overcome the EVD and SVD-related limitations is through incremental updates of existing EVD or SVD solutions according to new data. In that case, the full data set may not be available in advance, as in data streams. The solution obtained from a starting data point or data block is incrementally updated each time new data comes in. A recent study by Cardot and Degras (2017) reviewed a series of incremental eigenvalue decomposition approaches, namely, perturbation techniques, stochastic approximation and heuristic EVD/SVD. By exploiting these methods, the authors defined several incremental

PCA algorithms and compared their efficiency in terms of computational time and statistical accuracy. Their experiments indicated that perturbation techniques are relatively slow and require computing all eigenelements of the covariance matrix at each update; this makes their application prohibitive when dimensionality is high. Stochastic approximation methods provide almost sure convergence and have the highest computational speed, but they can generally process only one vector (observation) at a time. Finally, heuristic EVD/SVD approaches are less effficient in terms of speed but highly accurate; their asymptotic convergence has yet to be established.

Since MCA can be considered as a particular case of PCA, the aforementioned techniques can be interchangeably used to provide incremental MCA algorithms, i.e. sequential updates of existing MCA solutions. In contrast to incremental PCA, incremental MCA additionally involves the approximation of the column margins of the full indicator matrix, which, in online settings, is not available in advance. In that sense, an incremental MCA solution will always be approximate (Iodice D'Enza and Markos, 2015). Early work on this topic has been carried out by Benzécri (1969), who proposed a stochastic approximation algorithm for determining the largest eigenvalues of the expectation of a random matrix. Lebart (1974) provided a numerical proof of the convergence of Bénzecri's algorithm and indicated its potential in the case of sparse matrices, as those involved in MCA. Recently, Iodice D'Enza and Markos (2015) and Markos and Iodice D'Enza (2016) embedded an incremental SVD algorithm, emanating from the computer science literature, in the context of MCA and a related version of generalized canonical correlation analysis, and discussed its accuracy and convergence properties. However, a comparison of different methods that could lead to incremental MCA approaches is still missing. In this work we distinguish two main families of methods suitable for low-dimensional or low-rank incremental eigen-decomposition: stochastic approximation and heuristic incremental EVD/SVD. A general algorithmic framework is introduced to embed these methods in the MCA context and provide incremental updates of the solution. A comparison of competing methods on artificial data provides guidance on selecting a method in practice.

The paper is organized as follows: Section 2 presents PCA and MCA as matrix decomposition techniques. Section 3 briefly presents the most representative methods of stochastic approximation and heuristic incremental EVD/SVD. Based on these methods, a general framework for incrementally computing the MCA solution is introduced in Section 4. In Section 5, we present experimental results on

simulated data to investigate the accuracy of each method with regard to ordinary MCA. Section 6 gives conclusions and directions for future work.

2. DIMENSION REDUCTION METHODS AS A MATRIX DECOMPOSITION

This section provides a brief introduction to PCA and MCA from a matrix decomposition perspective. Let $\mathbf{X}$ be a $n \times Q$ data matrix, where n is the number of observations and Q is the number of quantitative attributes. For the sake of simplicity, we consider the Q attributes to be equally scaled. Then the PCA of $\mathbf{X}$ amounts the SVD of the following matrix

$$\mathbf{S}_{PCA} = n^{-1/2}\left(\mathbf{X} - n^{-1}\mathbf{1}\mathbf{1}^{\mathsf{T}}\mathbf{X}\right)Q^{-1/2}, \tag{1}$$

where $\mathbf{1}$ is an n-dimensional vector of 1. The decomposition is $\mathbf{S}_{PCA} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathsf{T}}$, where $\mathbf{U}$ is a $n \times Q$ orthonormal matrix with left singular vectors on columns and $\mathbf{W}$ is a diagonal matrix containing the Q singular values, $\mathbf{V}$ is a $Q \times Q$ matrix of right singular vectors. The j^{th} singular value corresponds to the standard deviation of data along the direction of j^{th} singular vector, $j = 1, \ldots, Q$. Let $\mathbf{U}_k$ and $\mathbf{V}_k$ be the first k columns of $\mathbf{U}$ and $\mathbf{V}$ and let $\mathbf{W}_k$ the matrix of the first k singular values, then $\mathbf{U}_k\mathbf{W}_k\mathbf{V}_k^{\mathsf{T}}$ is the rank k matrix that approximates $\mathbf{S}_{PCA}$ in the least-squares sense. The principal coordinates for rows and columns are $\mathbf{F} = n^{1/2}\mathbf{U}_k\mathbf{W}_k$ and $\mathbf{G} = Q^{1/2}\mathbf{V}_k\mathbf{W}_k$, respectively.

We define MCA in a very similar way: let $\mathbf{Z}$ be a $n \times Q$ binary matrix, where n is the number of observations and Q the total number of categories that characterize q categorical variables. The general element is $z_{ij} = 1$ if the i^{th} observation is characterized by the j^{th} category, $z_{ij} = 0$ otherwise; let $\mathbf{P} = \frac{1}{n \times q}\mathbf{Z}$ be the correspondence matrix, where $n \times q$ is the grand total of $\mathbf{Z}$. The core step of MCA is the matrix decomposition of the standardized residual matrix $\mathbf{S}_{\mathbf{Z}}$, defined as follows

$$\mathbf{S}_{\mathbf{Z}} = \mathbf{D}_{\mathbf{r}}^{-1/2}\left(\mathbf{P} - \mathbf{r}\mathbf{c}^{\mathsf{T}}\right)\mathbf{D}_{\mathbf{c}}^{-1/2}, \tag{2}$$

where $\mathbf{r}$ and $\mathbf{c}$ are the row and column margins of $\mathbf{P}$, respectively; $\mathbf{D}_{\mathbf{r}}$ and $\mathbf{D}_{\mathbf{c}}$ are diagonal matrices with values in $\mathbf{r}$ and $\mathbf{c}$. Consequently, the analogy with PCA is very close. As for PCA, MCA lies in the SVD of $\mathbf{S}_{MCA} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathsf{T}}$, and the principal coordinates on the k-dimensional space are $\mathbf{F} = \mathbf{D}_{\mathbf{r}}^{-1/2}\mathbf{U}_k\mathbf{W}_k$ for the observations, and $\mathbf{G} = \mathbf{D}_{\mathbf{c}}^{-1/2}\mathbf{V}_k\mathbf{W}_k$ for the attributes. Note that the MCA solution can be also obtained via the EVD of the standardized residuals of the Burt matrix $\mathbf{B} = \mathbf{Z}^{\mathsf{T}}\mathbf{Z}$, with $\mathbf{S}_{\mathbf{B}} = \mathbf{D}_{\mathbf{c}}^{-1/2}\left(\mathbf{P} - \mathbf{c}\mathbf{c}^{\mathsf{T}}\right)\mathbf{D}_{\mathbf{c}}^{-1/2}$.

3. MAIN APPROACHES TO LOW-RANK INCREMENTAL EIGEN-DECOMPOSITION

In this section we briefly present two main families of methods for incrementally computing the dominant eigenvalues and eigenvectors of a covariance matrix. The aim is to compute the principal eigenvector 'on the fly', without explicitly computing and storing the empirical covariance matrix.

3.1. STOCHASTIC APPROXIMATION

Stochastic approximation methods can be used to update the PCA solution incrementally by processing observations one-by-one. Benzécri (1969) and Krasulina (1970) have proposed independently stochastic approximation algorithms for determining the largest eigenvalues of the expectation of a random matrix. This method was extended by Oja and Karhunen (1985) and Oja (1992) with new proofs and developments, such as the Stochastic Gradient Ascent and Subspace Network Learning. Sanger (1989) and Weng et al. (2003) proposed similar approaches, Generalized Hebbian Algorithm and Candid Covariance-free Incremental PCA, respectively. More recently, Mitliagkas et al. (2013) introduced a block-wise stochastic variant of the classical power-method.

Stochastic Gradient Ascent - SGA (Oja, 1992). For a new random vector $\mathbf{x}_{n+1}$, the matrix $\mathbf{U}_n$ of orthonormal vectors is updated as follows:

$$\tilde{\mathbf{U}}_{n+1} = \mathbf{U}_n + \gamma_n(\mathbf{x}_{n+1} - \mu_{n+1})(\mathbf{x}_{n+1} - \mu_{n+1})^\top \mathbf{U}_n$$

$$\mathbf{U}_{n+1} = Orth(\tilde{\mathbf{U}}_{n+1}),$$

where *Orth()* is an orthonormalization step, e.g., using the Gram-Schmidt procedure. Notice that a tuning or gain parameter, γ_n, is involved which is often selected by trial-and-error.

Subspace Network Learning - SNL (Oja, 1992) is similar to SGA, where the orthonormalization step consists in multiplying $\tilde{\mathbf{U}}_{n+1}$ by $(\tilde{\mathbf{U}}_{n+1}^\top \tilde{\mathbf{U}}_{n+1}^{-1/2})$.

Generalized Hebbian Algorithm - GHA (Sanger, 1989). In this approach, the update step is given by

$$\mathbf{u}_{j,n+1} = \mathbf{u}_{j,n} + \gamma_n \phi_{j,n}\Big[(\mathbf{x}_{n+1} - \mu_{n+1}) - \phi_{j,n}\mathbf{u}_{j,n} - \Sigma_{i=1}^{j-1}\phi_{i,n}\mathbf{u}_{i,n}\Big],$$

where $\phi_{j,n} = (\mathbf{x}_{n+1} - \mu_{n+1})^\top \mathbf{u}_{j,n}$.

The asymptotic convergence of SGA, SNL and GHA is guaranteed and they are very efficient in terms of computational speed (Cardot and Degras, 2017). However, they only allow for vector updates, i.e., one data point at a time, and also require the choice of the turning parameter γ_n. No universally good values exist, but a usual choice is $\gamma_n = cn^{-\alpha}$ with $\alpha \in (1/2, 1)$ and $c \in (0, 0.8)$ (Cardot and Degras, 2017).

Candid Covariance-free Incremental PCA - CCI (Weng et al., 2003). CCI produces a sequence of stochastic approximations to the eigenvectors of $\mathbf{X}$ and then averages them. No tuning parameteters are required. Let $\mathbf{u}$ be an eigenvector of the covariance matrix with unit norm and let λ be the associated eigenvalue. Assume that estimates $\mathbf{v}_0, ..., \mathbf{v}_{n-1}$ of $\mathbf{v} = \lambda\mathbf{u}$ have been constructed in previous steps. Then $\mathbf{v}_{n+1}$ is obtained by:

$$\mathbf{v}_{n+1} = \frac{n}{n+1}\mathbf{v}_n + \frac{1}{n+1}\mathbf{x}_{n+1}\mathbf{x}_{n+1}^\top \frac{\mathbf{v}_n}{\| \mathbf{v}_n \|}.$$

The normalized eigenvector $\mathbf{u}$ and eigenvalue λ are estimated by $\mathbf{u}_n = \mathbf{v}_n / \| \mathbf{v}_n \|$ and $\lambda = \| \mathbf{v}_n \|$.

Block-wise Stochastic Power Method - BSP (Mitliagkas et al., 2013). This is a block-wise stochastic variant of the classical power-method with a variance reduction step. For each incoming block of size r, the empirical covariance matrix is multiplied by the matrix of orthonormal vectors $\mathbf{U}_n$, followed by an orthonormalization step:

$$\mathbf{S}_{n+1} = \frac{1}{r}\mathbf{x}_n\mathbf{x}_n^\top\mathbf{U}_n$$

$$\mathbf{U}_{n+1} = Orth(\mathbf{S}_{n+1})\,.$$

The optimal block size is $r \approx log(Q)/n$ (Mitliagkas et al., 2013).

3.2. HEURISTIC INCREMENTAL EVD/SVD

Heuristic approaches are based on the relationship between the SVD and the QR decomposition of a matrix (Arora et al., 2012; Levy and Lindenbaum, 2000; Ross et al., 2008). They allow for both vector- and block-based updates. Although they lead to highly accurate solutions, they are less efficient in terms of computational time and no theoretical guarantees exist for their performance (Cardot and Degras, 2017). This section describes two well-known approaches, mostly in the CS literature.

(Block-wise) Incremental PCA - (B)IPCA (Arora et al., 2012). The centered vector $\tilde{\mathbf{x}}_{n+1} = \mathbf{x}_{n+1} - \mu_n$ is decomposed as $\tilde{\mathbf{x}}_{n+1} = \mathbf{U}_n\mathbf{c}_{n+1} + \tilde{\mathbf{x}}^{\perp}_{n+1}$, where $\mathbf{c}_{n+1} = \mathbf{U}_n^{\mathsf{T}}\tilde{\mathbf{x}}_{n+1}$ are the coordinates of $\mathbf{x}_{n+1}$ in the k-dimensional space spanned by $\mathbf{U}_n$ and $\tilde{\mathbf{x}}^{\perp}_{n+1}$ is the projection of $\tilde{\mathbf{x}}_{n+1}$ onto the orthogonal space of $\mathbf{U}_n$.

Obtain the EVD of $\mathbf{Q}_{n+1} = \mathbf{V}_{n+1}\mathbf{W}_{n+1}\mathbf{V}_{n+1}^{\mathsf{T}}$ where

$$\mathbf{Q}_{n+1} = \frac{n}{(n+1)^2}\begin{pmatrix}(n+1)\mathbf{D}_n + \mathbf{c}_{n+1}\mathbf{c}_{n+1}^{\mathsf{T}} & \| \tilde{\mathbf{x}}^{\perp}_{n+1} \| \mathbf{c}_{n+1} \\ \| \tilde{\mathbf{x}}^{\perp}_{n+1} \| \mathbf{c}_{n+1}^{\mathsf{T}} & \| \tilde{\mathbf{x}}^{\perp}_{n+1} \|^2\end{pmatrix}$$

and $\mathbf{U}_{n+1} = \begin{pmatrix}\mathbf{U}_n & \frac{\mathbf{x}^{\perp}_{n+1}}{\|\mathbf{x}^{\perp}_{n+1}\|}\end{pmatrix}\mathbf{V}_{n+1}$ and $\mathbf{D}_{n+1} = \mathbf{W}_{n+1}$.
An extension of this method to block-wise updates is straightforward and is described in Levy and Lindenbaum (2000).
Block-wise Incremental SVD with mean update - BISVD (Ross et al., 2008). This is an extension of the previous method to account for the data mean. Given the SVD of $\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}\mu_x^{\mathsf{T}} = \mathbf{U}_x\mathbf{W}_x\mathbf{V}_x^{\mathsf{T}}$ of a starting data block $\mathbf{X}$, as well as the mean vector μ_y of an incoming data block $\mathbf{Y}$, the aim is to compute the SVD of $\begin{bmatrix}\tilde{\mathbf{X}} \\ \tilde{\mathbf{Y}}\end{bmatrix}$.

- Compute the **QR**-decomposition of $\mathbf{H} = \hat{\mathbf{Y}} - \mathbf{L}\mathbf{V}_x^{\mathsf{T}}$ and $\mathbf{K} = \begin{bmatrix}\mathbf{W}_x & 0 \\ \mathbf{L} & \mathbf{Q}\end{bmatrix}$, where $\mathbf{L} = \hat{\mathbf{Y}}\mathbf{V}_x$.
- Obtain the SVD of $\mathbf{K} = \hat{\mathbf{U}}\hat{\mathbf{W}}\hat{\mathbf{V}}$.
- Finally $\mathbf{U}_{xy} = \begin{bmatrix}\mathbf{U}_x & 0 \\ 0 & \mathbf{I}\end{bmatrix}\hat{\mathbf{U}}$, $\mathbf{W}_{xy} = \mathbf{W}$ and $\mathbf{V}_{xy} = \begin{bmatrix}\mathbf{V}_x & \mathbf{Q}\end{bmatrix}\hat{\mathbf{V}}$.
- Update $n_x = n_x + n_y$ and $\mu_x = \frac{n_x\mu_x + n_y\mu_y}{n_{xy}}$.

Note that the time required to compute the SVD of $\mathbf{K}$, which is approximately of size $n_y \times k$, does not depend on n_x, the number of units in $\mathbf{X}$. BISVD has been revised and embedded in the context of MCA by Iodice D'Enza and Markos (2015).

All methods described above are implemented in the `R` package `onlinePCA` (Cardot and Degras, 2016), with the exception of BISVD, which is implemented in the `R` package `idm` (Iodice D'Enza et al., 2017).

4. AN INCREMENTAL MCA FRAMEWORK

In order to adapt the incremental eigen-decomposition methods of the previous section in the context of MCA, we can express the covariance matrix involved in

the PCA computation

$$\frac{1}{n}\left(\mathbf{X}-\mathbf{1}\mu^{\mathsf{T}}\right)^{\mathsf{T}}\left(\mathbf{X}-\mathbf{1}\mu^{\mathsf{T}}\right)=\mathbf{U}\mathbf{W}\mathbf{U}^{\mathsf{T}}, \tag{3}$$

in terms of the indicator matrix $\mathbf{Z}$ and the Burt matrix $\mathbf{B}$ or, in other words, we can define the 'qualitative analogue' of $\mathbf{X}$ and the data mean, μ, which is the input of the incremental method. Note that μ is important in order to keep track of the data mean, which is a desirable property in the context of MCA, so as to simultaneously update the center of the low dimensional space of the solution (Iodice D'Enza and Markos, 2015).
First, the standardized residuals of the Burt matrix, $\mathbf{S_B}$, can be expressed in a form closer to Equation 3, as follows:

$$\begin{aligned}\mathbf{S_B}&=\mathbf{D_c}^{-1/2}(\mathbf{P_B}-\mathbf{c}\mathbf{c}^{\mathsf{T}})\mathbf{D_c}^{-1/2}=\mathbf{D_c}^{-1/2}\left(\frac{\mathbf{Z}^{\mathsf{T}}\mathbf{Z}}{nQ^2}-\mathbf{D_c}\mathbf{1_c}\mathbf{1_c}^{\mathsf{T}}\mathbf{D_c}\right)\mathbf{D_c}^{-1/2}=\\&=\left[\sqrt{n}\left(\frac{\mathbf{Z}}{Qn}-\frac{1}{n}\mathbf{1}_n\mathbf{1_c}^{\mathsf{T}}\mathbf{D_c}\right)\mathbf{D_c}^{-1/2}\right]^{\mathsf{T}}\left[\sqrt{n}\left(\frac{\mathbf{Z}}{Qn}-\frac{1}{n}\mathbf{1}_n\mathbf{1_c}^{\mathsf{T}}\mathbf{D_c}\right)\mathbf{D_c}^{-1/2}\right]\end{aligned} \tag{4}$$

where $\mathbf{1}_n$ and $\mathbf{1_c}$ are two vectors with n and c elements, respectively, each one equal to 1. Then, each of the elements of the product in Equation 4, can be written as follows:

$$\begin{aligned}&\sqrt{n}\left(\frac{\mathbf{Z}}{Qn}-\frac{1}{n}\mathbf{1}_n\mathbf{1_c}^{\mathsf{T}}\mathbf{D_c}\right)\mathbf{D_c}^{-1/2}=\\&\sqrt{n}\frac{\mathbf{Z}}{Qn}\mathbf{D_c}^{-1/2}-\frac{\sqrt{n}}{n}\mathbf{1}_n\mathbf{1_c}^{\mathsf{T}}\mathbf{D_c}\mathbf{D_c}^{-1/2}=\\&\frac{\mathbf{Z}}{Q\sqrt{n}}\mathbf{D_c}^{-1/2}-\frac{1}{\sqrt{n}}\mathbf{1}_n\mathbf{1_c}^{\mathsf{T}}\mathbf{D_c}^{1/2}=\\&\underbrace{\frac{\mathbf{Z}}{Q\sqrt{n}}\mathbf{D_c}^{-1/2}}_{\mathbf{X}}-\mathbf{1}_n\underbrace{\frac{1}{\sqrt{n}}\mathbf{1_c}^{\mathsf{T}}\mathbf{D_c}^{1/2}}_{\mu^{\mathsf{T}}}=\mathbf{S_Z}\end{aligned}$$

where $\mathbf{S_Z}$ is the standardized residual version of $\mathbf{Z}$. Therefore, the required quantities in the case of MCA are: $\mathbf{X}=\frac{1}{Q\sqrt{n}}\mathbf{Z}\mathbf{D_c}^{-1/2}$ and $\mu=\frac{1}{\sqrt{n}}\mathbf{D_c}^{1/2}\mathbf{1}$. Using the above definitions of $\mathbf{X}$ and μ, we can describe any incremental MCA algorithm.

Algorithm 1 A general Incremental MCA implementation

Require: $\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \\ \vdots \end{bmatrix}$ {incoming data stream split into blocks, with Q variables}

1: $\mathbf{Z}_x = \mathbf{Z}_1$ and $m = 1$
2: $\mathbf{S}_x = Q^{-1} n_x^{-1/2} \mathbf{Z}_x \mathbf{D}_{\mathbf{c}_x}^{1/2} - \mathbf{1}_{\mathbf{c}} n_x^{-1/2} \mathbf{1}_{\mathbf{c}}^{\top} \mathbf{D}_{\mathbf{c}_x}^{1/2}$ {standardized residuals}
3: $\mu_x = n_x^{-1/2} \mathbf{D}_{\mathbf{c}_x}^{1/2} \mathbf{1}_c$ {mean vector}
4: $\mathbf{S}_x = \mathbf{U}_x \mathbf{W}_x \mathbf{V}_x$ {starting eigenspace}
5: **while** (incoming data block) **do**
6: $\quad \mathbf{Z}_y = \mathbf{Z}_{m+1}$ {incoming block}
7: $\quad \mu_y = n_y^{-1/2} \mathbf{D}_{\mathbf{c}_y}^{1/2} \mathbf{1}_{\mathbf{c}}$ {mean vector}
8: $\quad n_{xy} = n_x + n_y$ {data size update}
9: $\quad \mathbf{D}_{\mathbf{r}_{xy}} = n_{xy}^{-1}$
10: $\quad \mathbf{D}_{\mathbf{c}_{xy}} = \left(n_x \mathbf{D}_{\mathbf{c}_x} + n_y \mathbf{D}_{\mathbf{c}_y} \right) n_{xy}^{-1}$ {margins update}
11: $\quad \mathbf{S}_y = Q^{-1} n_y^{-1/2} \mathbf{Z}_y \mathbf{D}_{\mathbf{c}_{xy}}^{-1/2} - \mathbf{1}_{\mathbf{c}} n_y^{-1/2} \mathbf{1}_{\mathbf{c}}^{\top} \mathbf{D}_{\mathbf{c}_y} \mathbf{D}_{\mathbf{c}_{xy}}^{-1/2}$ {standardized residuals}
12: $\quad \mathbf{S}_y = \mathbf{U}_y \mathbf{W}_y \mathbf{V}_y$ {eigenspace}
13: $\quad \mu_{xy} = \left(\mu_x n_x + \mu_y n_y \right) n_{xy}^{-1}$ {mean vector update}
14: $\quad$ Obtain $\mathbf{U}_{xy} \mathbf{W}_{xy} \mathbf{V}_{xy}$ {eigenspace update}
15: $\quad \mathbf{F}_{xy} = \mathbf{D}_{\mathbf{r}_{xy}}^{-1/2} \mathbf{U}_{xy} \mathbf{W}_{xy}$ {row principal coordinates}
16: $\quad \mathbf{G}_{xy} = \mathbf{D}_{\mathbf{c}_{xy}}^{-1/2} \mathbf{V}_{xy} \mathbf{W}_{xy}$ {column principal coordinates}
17: $\quad n_x = n_{xy}, \mu_x = \mu_{xy}, \mathbf{D}_{\mathbf{c}_x} = \mathbf{D}_{\mathbf{c}_{xy}}, \mathbf{U}_x = \mathbf{U}_{xy}, \mathbf{W}_x = \mathbf{W}_{xy}, \mathbf{V}_x = \mathbf{V}_{xy}$ {update}
18: $\quad m = m + 1$
19: **end while**

Algorithm 1 presents the pseudo-code for the general incremental MCA implementation. A categorical data stream arrives into blocks in the form of indicator matrices, $\mathbf{Z}$. For the first block, the standardized residuals matrix and the corresponding eigenspace are computed (Lines 2 and 4). Then the block is discarded. The procedure is iterated $m-1$ times, where $m-1$ is the total number of incoming blocks, leading to m updates in total. The subscript 'x' refers to the current data block, whereas the subscript 'y' refers to the incoming block. The updated quantities are then indicated by the subscript 'xy'. Note that the procedure does not require to store any of the data blocks processed up to a certain point, except for the incoming block. The crucial eigenspace update step (Line 14) can be achieved using any of the methods described in the previous section. The time and space

complexity of the algorithm largely depend on this step. Lines 15 and 16 calculate the row and column coordinates which can be used to visualize the solution, usually in two dimensions.

Recall that MCA involves a weighted SVD and the whole data matrix is unknown in advance. Assuming no missing values, the row weights (or margins), $\mathbf{D_r}$, of each data block are all equal to $1/Q$ and equal to the 'global' row margins, i.e., those of the full indicator matrix. The column margins, $\mathbf{D_c}$, however, are expected to differ across blocks and need to be approximated. A convenient choice for the global column margins is the average of the 'local' column margins, that is, the average margins of the data analyzed insofar (Line 10). The motivation for this choice and an investigation of the convergence properties as the number of data points increases is provided in Iodice D'Enza and Markos (2015).

5. SIMULATION STUDY

A simulation study was conducted to compare the performance of eight alternative incremental MCA variants, based on five stochastic approximation and three heuristic EVD/SVD approaches: Stochastic Gradient Ascent (SGA), Subspace Network Learning (SNL), Generalized Hebbian Algorithm (GHA), Candid Covariance-free Incremental PCA (CCI), Block Stochastic Power Method (BSP), Incremental PCA (IPCA), Block-wise Incremental PCA (BIPCA) and Block-wise Incremental SVD with mean update (BISVD). With the exception of BISVD, a thorough comparison in terms of time and space complexity of these methods is provided in Cardot and Degras (2017). The results are not expected to be any different when the methods are embedded in MCA. For this reason, we focus here our attention on a comparison in terms of accuracy, defined as the similarity between the ordinary MCA and the incremental MCA configurations in k dimensions. Another important aspect in this comparison is the performance of block-based methods (BSP, BIPCA and BISVD) against vector-based methods. Thus, we additionally investigate whether block-wise incremental analysis has any effect on accuracy.

The setup of the simulation study has been adopted from Cardot and Degras (2017). The function `poLCA.simdata` of the `R` package `poLCA` (Linzer and Lewis, 2011) was used to simulate categorical data sets that match the data-generating process assumed by the basic latent class model. The number of latent classes could randomly vary from 2 to 8 and the number of categories per variable from 2 to 7, with randomly generated probabilities of occurence of the different categories. Following Cardot and Degras (2017), the number of observations, n, was

generated with $n \in \{10^3, 10^4, 10^5, 10^6\}$ and the number of variables $Q \in \{10, 100\}$. The eight algorithms under comparison were initialized by an MCA of the first 25% of the observations and then run on the remaining 75%. The number of estimated dimensions, k, was fixed to 5. The tuning parameters of the stochastic methods, SGA, SNL and GHA, were selected by trial-and-error, as described in Cardot and Degras (2017). For block-based methods (BIPCA, BSP and BISVD), the recommended optimal block size was used (Iodice D'Enza and Markos, 2015; Levy and Lindenbaum, 2000; Mitliagkas et al., 2013).

The accuracy of each method was measured in terms of the similarity between the ordinary MCA and the incremental MCA configuration in principal coordinates. In particular, the similarity measure adopted was the R index, that equals $1 - m^2$, where m^2 is the symmetric orthogonal Procrustes statistic. The index ranges from 0 to 1 and can be interpreted as a correlation coefficient; it was calculated using the function `protest` of the package `vegan` (Oksanen et al., 2016). Ordinary MCA was applied using the `ca` package (Nenadic and Greenacre, 2007).

Table 1 shows the values of the R index (averaged over 1,000 replications) between ordinary MCA and each one of the eight incremental MCA variants under comparison, using the first $k = 5$ dimensions and different values of n and Q. First, we notice that as the number of observations increases from 10^3 to 10^6, incremental MCA configurations are getting more similar to those of ordinary MCA for all methods. Also, as expected, when the number of variables increases the performance generally deteriorates, since the number of retained dimensions is fixed. Stochastic approximation methods perform quite similar to each other, with the exception of BSP, which generally provides more accurate solutions. It is easy to observe that the three block-based algorithms, BSP, BIPCA and BISVD, lead to configurations that are more similar to ordinary MCA than vector-based methods. BISVD seems to outperform every other method with the exception of BSP, with which they have similar performance as n gets larger. It is important to outline, however, that block-based methods are expected to be less efficient than vector-based approaches in terms of computation time.

In incremental settings, retaining the very first few dimensions of the eigenspace being updated leads to reduced memory requirements and an increased overall efficiency of the updating process. Therefore, we investigated the effect of k, the number of retained dimensions during the updating process, on the similarity between ordinary and incremental MCA. The results are depicted in Figure 1 for 1,000 datasets of size $10^3 \times 100$, where the effect is more prominent. The average

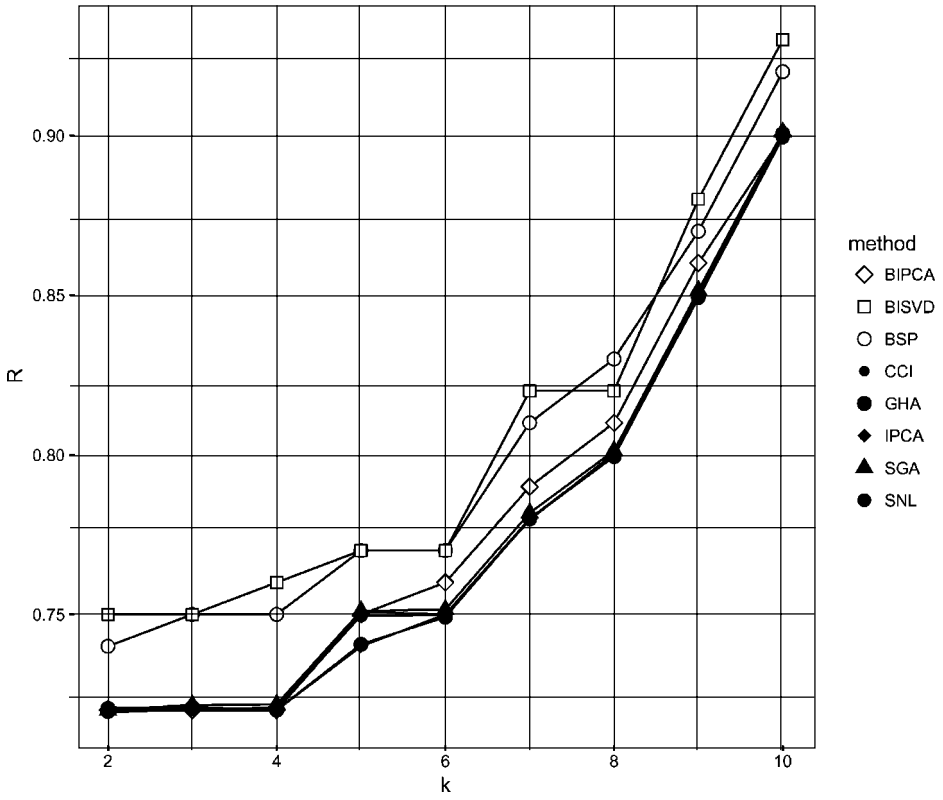

Figure 1: Similarity between ordinary and incremental MCA configurations obtained with eight different algorithms for varying number of retained dimensions (k = 2 to 10)

R index, plotted on the vertical axis, shows how similar the k-dimensional configuration of incremental MCA is to the corresponding ordinary MCA configuration. The degree of similarity was assessed for 2 up to 10 retained dimensions (horizontal axis). The eight methods are represented by different lines. As expected, incremental MCA becomes more similar to ordinary MCA with increasing dimensions, whereas the relative differences between methods seem to remain stable. It is important to highlight that even for $k = 2$ the configurations are highly similar.

Table 1: Similarity between ordinary MCA and incremental MCA configurations obtained with eight different algorithms for varying data size

data size	value	BSP	IPCA	BIPCA	GHA	CCI	SGA	SNL	BISVD
$10^3 \times 10$	mean	0.77	0.75	0.76	0.75	0.75	0.75	0.75	0.77
	sd	0.05	0.06	0.06	0.06	0.06	0.06	0.06	0.06
$10^3 \times 100$	mean	0.76	0.72	0.75	0.73	0.72	0.72	0.72	0.76
	sd	0.05	0.06	0.06	0.06	0.06	0.06	0.06	0.06
$10^4 \times 10$	mean	0.83	0.80	0.83	0.81	0.80	0.81	0.81	0.83
	sd	0.07	0.06	0.06	0.06	0.06	0.06	0.06	0.06
$10^4 \times 100$	mean	0.81	0.78	0.81	0.79	0.78	0.78	0.79	0.82
	sd	0.02	0.07	0.06	0.07	0.06	0.07	0.07	0.07
$10^5 \times 10$	mean	0.95	0.94	0.95	0.94	0.94	0.94	0.94	0.95
	sd	0.08	0.09	0.09	0.09	0.09	0.09	0.08	0.09
$10^5 \times 100$	mean	0.93	0.92	0.93	0.92	0.92	0.92	0.92	0.93
	sd	0.02	0.04	0.04	0.04	0.04	0.04	0.04	0.04
$10^6 \times 10$	mean	0.97	0.94	0.96	0.94	0.94	0.94	0.94	0.97
	sd	0.03	0.04	0.04	0.04	0.04	0.04	0.04	0.04
$10^6 \times 100$	mean	0.97	0.94	0.95	0.93	0.93	0.94	0.93	0.96
	sd	0.04	0.05	0.05	0.05	0.05	0.05	0.05	0.06

6. CONCLUSIONS

In this paper we presented a general framework for computing low-rank incremental MCA solutions. By exploiting the relationship between PCA and MCA, we embedded two main families of incremental PCA to the MCA context. A series of experiments indicated that block-wise approaches, BISVD, BSP and BIPCA, outperform vector-based methods in terms of computational accuracy; it seems that larger incoming data blocks tend to reduce noise and estimation variability. In addition, block-based methods offer a good compromise between accuracy and speed. However, no theoretical guarantees exist (yet) for their performance. On the contrary, in line with Cardot and Degras (2017), vector-based stochastic approaches, SGA, SNL, GHA and CCI, provide the highest computation speed and their almost sure convergence has been established. In practice, the best method should be chosen on the basis of the most desirable balance of speed and accuracy, given the circumstances.

Future research could focus on at least three areas. First, how incremental approaches accommodate nonstationary processes, i.e., how they account for changes in the data streams needs further investigation. The stochastic algorithms of Section 3.1 naturally accommodate nonstationary processes through the tuning parameter γ. This parameter controls the weight given to earlier observations. In the case of heuristic approaches, a so-called forgetting factor can be incorporated

in the decomposition to down-weight older data points; its effect on accuracy of the solution needs to be evaluated. Second, since missing values in data streams is a crucial issue in many real world applications, efficient strategies are needed which do not impose a considerable increase in computation time or a decrease in accuracy. Third, incremental versions of more general multivariate analysis methods based on the EVD/SVD, such as Multiple Factor Analysis and Generalized Canonical Correlation Analysis, is also worth further investigation.

REFERENCES

Arora, R., Cotter, A., Livescu, K. and Srebro, N. (2012). Stochastic optimization for PCA and PLS. In 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 861–868.

Benzécri, J.P. (1969). Approximation stochastique dans une algèbre normée non commutative. In *Bulletin de la Societé Mathematique de France*, 97: 225–241.

Cardot, H. and Degras, D. (2016). *onlinePCA: Online Principal Component Analysis.* URL http://CRAN.R-project.org/package=onlinePCA. R package version 1.3.1.

Cardot, H. and Degras, D. (2017). Online principal component analysis in high dimension: Which algorithm to choose? In *International Statistical Review (forthcoming).*

Greenacre, M. (2007). *Correspondence Analysis in Practice.* London: CRC Press.

Iodice D'Enza, A. and Markos, A. (2015). Low-dimensional tracking of association structures in categorical data. In *Statistics and Computing,* 25 (5): 1009– 1022.

Iodice D'Enza, A., Markos, A. and Buttarazzi, D. (2017). idm: Incremental Decomposition Methods in R. In *Journal of Statistical Software (forthcoming).*

Krasulina, T. (1970). Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices. In *Automation and Remote Control,* 2: 50–56.

Lebart, L. (1974). On the Benzécri's method for finding eigenvectors by stochastic approximation. In *COMPSTAT, Proceedings in Computational Statistics,* 202– 211.

Levy, A. and Lindenbaum, M. (2000). Sequential Karhunen-Loeve basis extraction and its application to images. In *IEEE Transactions on Image Processing,* 9 (8): 1371–1374.

Linzer, D.A. and Lewis, J.B. (2011). poLCA: An R package for polytomous variable latent class analysis. In *Journal of Statistical Software,* 42 (10): 1–29.

Markos, A. and Iodice D'Enza, A. (2016). Incremental generalized canonical correlation analysis. In F.A.Wilhelm and A.H. Kestler, eds., *Analysis of Large and Complex Data,* 185–194. Springer International Publishing.

Mitliagkas, I., Caramanis, C. and Jain, P. (2013). Memory limited, streaming PCA. In C.J.C. Burges, L. Bottou, M.Welling, Z. Ghahramani, and K.Q.Weinberger, eds., *Advances in Neural Information Processing Systems 26,* 2886– 2894. Curran Associates, Inc.

Nenadic, O. and Greenacre, M. (2007). Correspondence Analysis in R, with twoand three-dimensional graphics: The ca package. In *Journal of Statistical Software,* 20 (3): 1–13.

Oja, E. (1992). Principal components, minor components, and linear neural networks. In *Neural Networks,* 5 (6): 927–935.

Oja, E. and Karhunen, J. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. In *Journal of Mathematical Analysis and Applications,* 106 (1): 69–84.

Oksanen, J., Blanchet, F.G., Friendly, M., Kindt, R., Legendre, P., McGlinn, D., Minchin, P.R., O'Hara, R.B., Simpson, G.L., Solymos, P., Stevens, M.H.H., Szoecs, E. and Wagner, H. (2016). *vegan: Community Ecology Package.* URL `https://CRAN.R-project.org/package=vegan`. R package version 2.4-0.

Ross, D.A., Lim, J., Lin, R.S. and Yang, M.H. (2008). Incremental learning for robust visual tracking. In *International Journal of Computer Vision,* 77 (1-3): 125–141.

Sanger, T.D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. In *Neural Networks,* 2 (6): 459–473.

Weng, J., Zhang, Y. and Hwang, W.S. (2003). Candid covariance-free incremental principal component analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 25 (8): 1034–1040.