Incremental Generalized Canonical Correlation Analysis

Angelos Markos and Alfonso Iodice D'Enza

Abstract Generalized canonical correlation analysis (GCANO) is a versatile technique that allows the joint analysis of several sets of data matrices through data reduction. The method embraces a number of representative techniques of multivariate data analysis as special cases. The GCANO solution can be obtained noniteratively through an eigenequation and distributional assumptions are not required. The high computational and memory requirements of ordinary eigendecomposition makes its application impractical on massive or sequential data sets. The aim of the present contribution is twofold: (a) to extend the family of GCANO techniques to a split-apply-combine framework, that leads to an exact implementation; (b) to allow for incremental updates of existing solutions, which lead to approximate yet highly accurate solutions. For this purpose, an incremental SVD approach with desirable properties is revised and embedded in the context of GCANO, and extends its applicability to modern big data problems and data streams.

1 Introduction

Numerous procedures for relating multiple sets of variables have been described in the literature (Van der Burg 1988; Gifi 1990; Kroonenberg 2008). In this paper, we consider a generalized version of canonical correlation analysis (GCANO) developed by Carroll (1968). The central problem of GCANO is to construct a series of components, or canonical variates, aiming to maximize the association or homogeneity among the multiple variable sets. This version is most attractive because the solution can be obtained through an eigenequation, strict distributional

A. Markos (✉)
Department of Primary Education, Democritus University of Thrace, Xanthi, Greece
e-mail: amarkos@eled.duth.gr; amarkos@gmail.com

A.I. D'Enza
Department of Economics and Law, Università di Cassino e del Lazio Meridionale, Cassino FR, Italy
e-mail: iodicede@unicas.it; iodicede@gmail.com

A.F.X. Wilhelm, H.A. Kestler (eds.), *Analysis of Large and Complex Data*, Studies in Classification, Data Analysis, and Knowledge Organization,
DOI 10.1007/978-3-319-25226-1_16

assumptions are not required and, most importantly, the method subsumes a number of representative techniques of multivariate data analysis as special cases (Takane et al. 2008; Van de Velden and Takane 2012). In the case of two data sets with continuous variables, GCANO reduces to canonical correlation analysis. When one of the two sets of variables consists of indicator variables, the method specializes into canonical discriminant analysis and into correspondence analysis when both sets consist of indicator variables. In the case of more than two data sets with indicator variables, GCANO specializes into multiple correspondence analysis, and into principal component analysis when each of the data sets consists of a single continuous variable. Therefore, a useful modification of the GCANO algorithm has far reaching implications beyond what is normally referred to as GCANO (Takane et al. 2008).

GCANO has been profitably applied as a tool for integrating data obtained from different sources (e.g., subjects, stimuli, locations), in fields ranging from marketing (Bijmolt and Van de Velden 2012) to neuroimaging (Correa et al. 2010). In the last few years, new application frameworks emerged that usually involve large/massive amounts of data. Some of the examples that can be given in this context are the continuous monitoring of consumer preferences plotted on perceptual maps, fusing data concurrently acquired from different imaging modalities, and monitoring of word associations that are present in data pulled on-the-fly from social networking sites. In all 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 and require fast response time and efficient memory use.

In fact, the capability of GCANO is challenged by such data and requires a different approach. The problem is that finding the (generalized) canonical correlations requires a computationally expensive eigendecomposition. More specifically, the application of ordinary eigenvalue decomposition (EVD) or singular value decomposition (SVD) to large and high-dimensional data becomes infeasible because of high computational and memory requirements. This subsequently makes the application of GCANO impractical on massive or sequential data, 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.

Literature offers several proposals aiming to overcome the EVD and SVD-related limitations via efficient eigensolvers (e.g., Baglama and Reichel 2007) or via the update (or downdate) of existing EVD/SVD solutions according to new data (see Baker et al. 2012 for an overview). Another strategy for tackling large data problems is the so-called split-apply-combine (Wickam 2011): the full data set is split into blocks, each block is analyzed separately and the results are combined to obtain the global solution. In that case, the solution corresponding to the decomposition of the starting data block has to be incrementally updated each time new data comes in.

The aim of the present contribution is twofold: (a) to extend the family of GCANO techniques to a split-apply-combine framework, that leads to an exact solution, i.e. to the exact calculation of canonical correlations; (b) to allow for incremental updates of existing solutions, which lead to approximate yet highly

accurate solutions. For this purpose, an incremental SVD approach with desirable properties is revised and embedded in the context of GCANO, and extends its applicability to modern big data problems and data streams.

The paper is organized as follows: Sect. 2 presents GCANO as a matrix decomposition technique. Section 3 focuses on an incremental SVD approach with desirable properties in the context of GCANO. Based on this approach, two algorithmic modifications for incrementally computing the GCANO solution are proposed in Sect. 4. In Sect. 5 we illustrate a real-world application on data gathered from a social networking site. The paper concludes in Sect. 6.

2 Generalized Canonical Correlation Analysis

Let $\mathbf{Z}_k$ denote an n by p_k matrix of variables of the kth data set ($k = 1, \cdots K$), where n is the number of cases. We assume that $\mathbf{Z}_k$ is column-wise standardized. Let $\mathbf{Z}$ denote an n by $p = (\Sigma_k p_k)$ row block matrix, $\mathbf{Z} = [\mathbf{Z}_1, \ldots, \mathbf{Z}_K]$. Let $\mathbf{W}_k$ denote a p_k by d matrix of weights assigned to each variable in $\mathbf{Z}_k$, where d is the number of dimensions. Let $\mathbf{F}$ denote an n by d matrix of low dimensional data representations, known as object scores, which characterize the association or homogeneity among all $\mathbf{Z}_k$'s. The aim of GCANO is to obtain $\mathbf{W}$ which maximizes

$$\phi(\mathbf{W}) = \text{tr}(\mathbf{W}^\mathsf{T}\mathbf{Z}^\mathsf{T}\mathbf{Z}\mathbf{W}) \tag{1}$$

subject to the restriction that $\mathbf{W}^\mathsf{T}\mathbf{D}\mathbf{W} = \mathbf{I}_t$, where $\mathbf{D}$ is a p by p block diagonal matrix formed from $\mathbf{D}_k = \mathbf{Z}_k^\mathsf{T}\mathbf{Z}$ as the kth diagonal block.

The solution can be achieved through different algorithmic approaches (see Takane et al. 2008), but the most relevant one to our purpose is via the generalized singular value decomposition (GSVD) of matrix $\mathbf{Z}\mathbf{D}^-$ with column metric $\mathbf{D}$, where $\mathbf{D}^-$ is a generalized inverse of $\mathbf{D}$. In other words, GCANO can be performed as a principal component analysis applied to the global table $\mathbf{Z}$, with metric corresponding to the block diagonal matrix composed of the inverses of the variance-covariance matrices internal to every group $\mathbf{Z}_k$. This is equivalent to obtaining the ordinary SVD of:

$$\mathbf{Z}\mathbf{D}^{-1/2} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\mathsf{T}. \tag{2}$$

Then canonical weights are given by $\mathbf{W} = \mathbf{D}^{-1/2}\mathbf{V}$ and canonical variates by $\mathbf{F} = \mathbf{Z}\mathbf{W}\boldsymbol{\Sigma}^{1/2}$ (Takane et al. 2008).

A convenient choice of $\mathbf{D}^{-1/2}$ is $\mathbf{D}^-\mathbf{D}^{1/2}$, where $\mathbf{D}^-$ is an arbitrary g-inverse and $\mathbf{D}^{1/2}$ is the symmetric square root factor of $\mathbf{D}$. Different choices of $\mathbf{D}^{-1/2}$ lead to the different methods which lie under the GCANO framework. For instance, in the case of correspondence analysis $\mathbf{D}^{-1/2}$ is replaced by $(\mathbf{D}^+)^{1/2}$, where $\mathbf{D}^+$ is the Moore–Penrose inverse of $\mathbf{D}$ and each $\mathbf{Z}_k$ is a set of column-centered indicator matrices. In the case of principal component analysis, the solution is simply given by the SVD of $\mathbf{Z}$; thus $\mathbf{D}^{-1/2} = \mathbf{I}$.

3 Enhancing the SVD Computation

A downside of SVD and related EVD is their high computational cost. The SVD has a computational complexity of $\mathcal{O}(n^2p)$, assuming $n \geq p$ (Golub and Van Loan 1996). Therefore, both methods become computationally infeasible for large data sets. In the literature, we find two broad classes of methods which lead to efficient eigendecompositions.

The first class, known as batch methods, requires that all the data is available in advance to perform the decomposition. This class includes methods such as iterative EVD (Golub and Van Loan 1996) and bilinear diagonalization Lanczos (Baglama and Reichel 2007). Although these methods enable very efficient computations, their application is not possible in cases where the data size is too large to fit in memory, or if the full data set is not available in advance, as in the case of data streams. In the latter case, it may be advantageous to perform the computations as the data become available.

The so-called incremental methods can operate on streaming data and aim to update (or downdate) an existing SVD or EVD solution when new data is processed. This class of methods includes some expectation-maximization (e.g., Tipping and Bishop 1999), stochastic approximation (e.g., Herbster and Warmuth 2001), and sequential decomposition approaches (see Baker et al. 2012; Iodice D' Enza and Markos 2015, for an overview). These methods have an advantage over batch methods as they can be applied to sequential data blocks without the need to store past data in memory.

In this paper, we utilize the desirable properties of a sequential decomposition algorithm, (herein referred to as Incremental SVD) described by Ross et al. (2008), which is based on a series of efficient block updates instead of a full and expensive SVD. The procedure allows to keep track of the data mean, so as to simultaneously update the center of the low dimensional space of the solution. This property is important when the data sets consist of indicator variables, as in the case of correspondence analysis (Iodice D' Enza and Markos 2015). Second, the method offers a computational advantage over alternatives in that the decomposition can be computed in constant time regardless of data size. This property makes it more appealing for an incremental GCANO implementation in the case of data streams.

3.1 Incremental SVD

In this section, we present an algorithm which exploits the fact that a low-rank update to the eigenbasis is decomposable into efficient block operations. In the original description of the method (Ross et al. 2008), the data is updated column-wise, but here we derive a row-wise update formulation of the method. This is subsequently utilized in the following section to provide an incremental GCANO approach, suitable for analyzing data streams.

Before describing the core of the method, we introduce some necessary definitions. An *eigenspace* is a collection of the quantities needed to define the result of a matrix eigendecomposition as it involves eigenvalues (singular values), eigenvectors (singular vectors), data mean, and size. In particular, with respect to the SVD, for a $n^{(x)} \times p$ matrix $\mathbf{X}$ and a $n^{(y)} \times p$ matrix $\mathbf{Y}$, we define two eigenspaces as

$$\Omega^{(x)} = \left(n^{(x)}, \mu^{(x)}, \mathbf{U}^{(x)}, \boldsymbol{\Sigma}^{(x)}, \mathbf{V}^{(x)}\right) \quad \text{and} \quad \Omega^{(y)} = \left(n^{(y)}, \mu^{(y)}, \mathbf{U}^{(y)}, \boldsymbol{\Sigma}^{(y)}, \mathbf{V}^{(y)}\right).$$

The aim of incremental decomposition is to obtain an eigenspace $\Omega^{(xy)}$ for the matrix $\begin{bmatrix}\mathbf{X}\\ \mathbf{Y}\end{bmatrix}$, using uniquely the information in $\Omega^{(x)}$ and $\Omega^{(y)}$.

The total number of statistical units and the global data mean are easily updated: $n^{(xy)} = n^{(x)} + n^{(y)}$ and $\mu^{(xy)} = \frac{n^{(x)}\mu^{(x)} + n^{(y)}\mu^{(y)}}{n^{(xy)}}$. Adding the eigenspaces acts to rotate the eigenvectors (or singular vectors) and to scale the eigenvalues (or singular values) relating to data spread; furthermore, the new eigenvectors must be a linear combination of the old.

Let $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ be the centered versions of $\mathbf{X}$ and $\mathbf{Y}$, respectively. In order to take into account the varying mean, the row vector $\sqrt{\frac{n^{(x)}n^{(y)}}{n^{(x)}+n^{(y)}}}\left(\mu^{(y)} - \mu^{(x)}\right)$ is added to the $\tilde{\mathbf{Y}}$ matrix. Given the SVD of $\tilde{\mathbf{X}} = \mathbf{U}^{(x)}\boldsymbol{\Sigma}^{(x)}\left(\mathbf{V}^{(x)}\right)^{\mathsf{T}}$, the projection $\mathbf{L}$ of $\tilde{\mathbf{Y}}$ onto the orthogonal basis $\left(\mathbf{V}^{(x)}\right)^{\mathsf{T}}$ is described by:

$$\mathbf{L} = \tilde{\mathbf{Y}}\left(\mathbf{V}^{(x)}\right)^{\mathsf{T}}.$$

Let $\mathbf{H}$ be the component of $\tilde{\mathbf{Y}}$ orthogonal to the subspace spanned by $\left(\mathbf{V}^{(x)}\right)^{\mathsf{T}}$:

$$\mathbf{H} = \tilde{\mathbf{Y}}\left(\mathbf{I} - \left(\mathbf{V}^{(x)}\right)^{\mathsf{T}}\mathbf{V}^{(x)}\right) = \tilde{\mathbf{Y}} - \mathbf{L}\mathbf{V}^{(x)}.$$

$\mathbf{H}$ is decomposed such that an orthogonal matrix $\mathbf{V}^{(h)}$ is obtained, as follows:

$$\mathbf{V}^{(h)} = \operatorname{orth}\left(\tilde{\mathbf{Y}} - \mathbf{L}\mathbf{V}^{(x)}\right).$$

Then, $\begin{bmatrix}\tilde{\mathbf{X}}\\ \tilde{\mathbf{Y}}\end{bmatrix}$ is given by

$$\begin{bmatrix}\tilde{\mathbf{X}}\\ \tilde{\mathbf{Y}}\end{bmatrix} = \left(\begin{bmatrix}\mathbf{U}^{(x)} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\mathbf{U}^{(r)}\right)\boldsymbol{\Sigma}^{(r)}\left(\mathbf{V}^{(r)}\begin{bmatrix}\mathbf{V}^{(x)}\\ \mathbf{V}^{(h)}\end{bmatrix}\right), \tag{3}$$

where $\mathbf{U}^{(r)}\boldsymbol{\Sigma}^{(r)}\left(\mathbf{V}^{(r)}\right)^{\mathsf{T}}$ is the SVD of $\mathbf{R} = \begin{bmatrix}\boldsymbol{\Sigma}^{(x)} & \mathbf{0}\\ \mathbf{L} & \mathbf{H}\left(\mathbf{V}^{(h)}\right)^{\mathsf{T}}\end{bmatrix}$.

Finally, $\mathbf{U}^{(xy)} = \begin{bmatrix} \mathbf{U}^{(x)} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{U}^{(r)}$, $\boldsymbol{\Sigma}^{(xy)} = \boldsymbol{\Sigma}^{(r)}$ and $\mathbf{V}^{(xy)} = \mathbf{V}^{(r)} \begin{bmatrix} \mathbf{V}^{(x)} \\ \mathbf{V}^{(h)} \end{bmatrix}$.
It should be noted that in many practical situations, it suffices to obtain only $\mathbf{V}^{(xy)}$, the basis for the right singular subspace.

4 Dynamic Modifications of GCANO Solutions

This Section describes two dynamic modifications of the GCANO algorithm, referred to as "Exact" and "Live." The Exact approach is an incremental GCANO which leads to the same exact solution as the one obtained using ordinary GCANO. This approach requires all the data to be available from the start. On the other hand, the Live approach is suitable for analyzing data blocks incrementally as they arrive, but leads to an approximate GCANO solution. The main difference between the two approaches lies in the calculation of the block diagonal matrix $\mathbf{D}$. With regard to the Exact approach, its calculation is based on the "global" variance-covariance matrices internal to every data block, that is, the matrices which correspond to the whole data matrix, which is available in advance. For the Live approach, the whole matrix is unknown and the variance-covariance matrices internal to every data block are approximated by the "local" matrices, that is, the average variance-covariance matrices of the data analyzed insofar. A detailed description of the two implementations follows and the corresponding pseudo-code is provided in Algorithms 1 and 2.

4.1 Exact GCANO

Algorithm 1 summarizes the Exact approach. The procedure is iterated k times, where k is the total number of incoming blocks leading to k updates. The superscript "(x)" indicates a quantity referred to the current block, whereas the superscript "(y)" refers to the incoming block. The updated quantities are then indicated by the superscript "(xy)". The merging of eigenspaces is achieved using the SVD-based method described in Sect. 3.1.

In terms of time complexity, the Exact GCANO algorithm is data dependent and is expected to yield much lower space complexity than ordinary GCANO. More specifically, computing an eigenspace model of size $n \times p$ using the SVD, usually incurs a computational cost of $O(n^2p)$. Therefore, using the batch approach, the merging of two consecutive eigenspaces requires approximately $O((n^{(x)} + n^{(y)})^2p)$ operations. Now we focus on the parts which dominate the computational complexity of the Exact algorithm. The starting and the incoming eigenspaces (Steps 4 and 8) need a total of $O(((n^{(x)})^2 + (n^{(y)})^2)p)$ operations and eigenspace merging (Step 11) requires at most $O((d + n^{(y)} + 1)^2p)$. Note that d is the number of the largest singular values retained in each step.

Algorithm 1: Exact GCANO

Require: $\mathbf{Z} = [\mathbf{Z}_1, \cdots, \mathbf{Z}_K]$ {$n \times p$, K sets}
1: $\mathbf{D}[k,k] = \mathbf{Z}_k^{\mathsf{T}}\mathbf{Z}_k$ {form block-diagonal $\mathbf{D}$}
2: $\mathbf{X} = \mathbf{Z}\mathbf{D}^{-1/2} = [\mathbf{X}_1; \mathbf{X}_2; \cdots; \mathbf{X}_s]$ {calculate $\mathbf{X}$ and split into s blocks}
3: $\mathbf{X}^{(x)} = \mathbf{X}_1$
4: $\boldsymbol{\Omega}^{(x)} = (n^{(x)}, \mu^{(x)}, \mathbf{U}^{(x)}, \boldsymbol{\Sigma}^{(x)}, \mathbf{V}^{(x)})$ {starting eigenspace}
5: **while** (incoming data block) **do**
6: $\quad\mathbf{X}^{(y)} = \mathbf{X}_{m+1}$ {incoming block}
7: $\quad\mu^{(y)} = \left(n^{(y)}\right)^{-1}\left(\mathbf{X}^{(y)}\right)^{\mathsf{T}}\mathbf{1}_{n^{(y)}}$ {mean vector}
8: $\quad\boldsymbol{\Omega}^{(y)} = (n^{(y)}, \mu^{(y)}, \mathbf{U}^{(y)}, \boldsymbol{\Sigma}^{(y)}, \mathbf{V}^{(y)})$ {new eigenspace}
9: $\quad n^{(xy)} = n^{(x)} + n^{(y)}$ {data size update}
10: $\quad\mu^{(xy)} = \left[\mu^{(x)}n^{(x)} + \mu^{(y)}n^{(y)}\right]\left(n^{(x)}\right)^{-1}$ {mean vector update}
11: $\quad\boldsymbol{\Omega}^{(xy)} = \boldsymbol{\Omega}^{(x)} \oplus \boldsymbol{\Omega}^{(y)} = (n^{(xy)}, \mu^{(xy)}, \mathbf{U}^{(xy)}, \boldsymbol{\Sigma}^{(xy)}, \mathbf{V}^{(xy)})$ {eigenspace update}
12: $\quad\mathbf{W}^{(xy)} = \mathbf{D}^{-1/2}\mathbf{V}^{(xy)}$ {canonical weights}
13: $\quad\mathbf{F} = \mathbf{Z}\mathbf{W}^{(xy)}\left(\boldsymbol{\Sigma}^{(xy)}\right)^{-1/2}$ {canonical variates}
14: $\quad n^{(x)} = n^{(xy)}, \mu^{(x)} = \mu^{(xy)}, \mathbf{U}^{(x)} = \mathbf{U}^{(xy)}, \boldsymbol{\Sigma}^{(x)} = \boldsymbol{\Sigma}^{(xy)}, \mathbf{V}^{(x)} = \mathbf{V}^{(xy)}$ {update}
15: **end while**

4.2 Live GCANO

This section introduces the Live GCANO algorithm and discusses some of its basic properties, in comparison with the Exact case of the previous section. The pseudo-code for the Live case is presented in Algorithm 2. Since the data is not available from the beginning, k, the total number of blocks being analyzed, is not defined in advance. The most crucial difference between the Exact and Live approach lies in the computation of the block diagonal matrix $\mathbf{D}$ of the incoming block. In fact, an additional step (Step 7) is required within the loop of Algorithm 2, where the average of the "local" variance-covariance matrices is obtained.

The complexity of the Live algorithm can be summarized in a similar way to that of the Exact case. Each update (Step 13) requires approximately $O((d+n^{(y)}+1)^2p)$ flops, versus $O((n^{(x)}+n^{(y)})^2p)$ with the naive approach. Another important feature is that the decomposition can be computed in constant time by constraining the update block size (see Iodice D' Enza and Markos 2015 for details). The storage required for the $n^{(y)}$ new rows reduces significantly the space complexity to approximately $O((d+n^{(y)}+1)p)$, down from $O((n^{(x)}+n^{(y)})^2p)$ required for the naive SVD.

Algorithm 2: Live GCANO

Require: $\mathbf{Z} = [\mathbf{Z}_1; \mathbf{Z}_2; \cdots]$ {incoming data stream}

$\mathbf{Z}^{(x)} = \mathbf{Z}_1$

$\mathbf{D}^{(x)}[k,k] = \left(\mathbf{Z}_k^{(x)}\right)^{\mathsf{T}} \mathbf{Z}_k^{(x)}$ {form block-diagonal $\mathbf{D}_x$}

$\mathbf{X}^{(x)} = \mathbf{Z}^{(x)} \left(\mathbf{D}^{(x)}\right)^{-1/2}$ {calculate $\mathbf{X}^{(x)}$}

$\boldsymbol{\Omega}^{(x)} = (n^{(x)}, \mu^{(x)}, \mathbf{U}^{(x)}, \boldsymbol{\Sigma}^{(x)}, \mathbf{V}^{(x)})$ {starting eigenspace}

while (incoming data block) **do**

  $\mathbf{Z}^{(y)} = \mathbf{Z}_{m+1}$ {incoming block}

  $\mathbf{D}^{(xy)}[k,k] = \left(\left(\mathbf{Z}^{(x)}\right)^{\mathsf{T}} \mathbf{Z}^{(x)} + \left(\mathbf{Z}^{(y)}\right)^{\mathsf{T}} \mathbf{Z}^{(y)}\right) / n^{(x)}$ {update block-diagonal $\mathbf{D}$}

  $\mathbf{X}^{(y)} = \mathbf{Z}^{(y)} \left(\mathbf{D}^{(xy)}\right)^{-1/2}$ {calculate $\mathbf{X}^{(y)}$}

  $\mu^{(y)} = \left(n^{(y)}\right)^{-1} \left(\mathbf{X}^{(y)}\right)^{\mathsf{T}} \mathbf{1}_{n^{(y)}}$ {mean vector}

  $\boldsymbol{\Omega}^{(y)} = (n^{(y)}, \mu^{(y)}, \mathbf{U}^{(y)}, \boldsymbol{\Sigma}^{(y)}, \mathbf{V}^{(y)})$ {new eigenspace}

  $n^{(xy)} = n^{(x)} + n^{(y)}$ {data size update}

  $\mu^{(xy)} = \left[\mu^{(x)} n^{(x)} + \mu^{(y)} n^{(y)}\right] \left(n^{(x)}\right)^{-1}$ {mean vector update}

  $\boldsymbol{\Omega}^{(xy)} = \boldsymbol{\Omega}^{(x)} \oplus \boldsymbol{\Omega}^{(y)} = (n^{(xy)}, \mu^{(xy)}, \mathbf{U}^{(xy)}, \boldsymbol{\Sigma}^{(xy)}, \mathbf{V}^{(xy)})$ {eigenspace update}

  $\mathbf{W}^{(xy)} = \left(\mathbf{D}^{(xy)}\right)^{-1/2} \mathbf{V}^{(xy)}$ {canonical weights}

  $\mathbf{F}^{(xy)} = \mathbf{Z}^{(xy)} \mathbf{W}^{(xy)} \left(\boldsymbol{\Sigma}^{(xy)}\right)^{-1/2}$ {canonical variates}

  $n^{(x)} = n^{(xy)}, \mu^{(x)} = \mu^{(xy)}, \mathbf{U}_{(x)} = \mathbf{U}^{(xy)}, \boldsymbol{\Sigma}^{(x)} = \boldsymbol{\Sigma}^{(xy)}, \mathbf{V}^{(x)} = \mathbf{V}^{(xy)}, \mathbf{Z}^{(x)} = \mathbf{Z}^{(y)}$ {update}

end while

5 Application

The two proposed approaches were applied to a real-world data set. For convenience, we consider only sets of categorical variables, thus GCANO results coincide with those of multiple correspondence analysis (MCA). A detailed description of Algorithms 1 and 2 in the case of MCA can be found in Iodice D' Enza and Markos (2015). The data refers to a small corpus of messages or tweets mentioning seven major hotel brands. It was gathered by continuously querying and archiving the Twitter Streaming API service, which provides a proportion of the most recent publicly available tweets, along with information about the user. The data was collected using the twitteR package in R (Gentry 2011). A total of about 10,000 tweets in English were extracted within a time period of 6 consecutive days.

Apart from brand name, two additional variables were considered. A sentiment score was assigned to each tweet by first counting the number of occurrences of "positive" and "negative" words according to a sentiment dictionary and then subtracting the number of occurrences of negative words from the number of positive. Larger negative scores correspond to more negative expressions of sentiment, neutral (or balanced) tweets net to zero, and very positive tweets score larger, positive numbers. A sentiment polarity variable of either "positive +", "neutral +/−" or "negative −" sentiment was finally obtained by simply taking the sign of the sentiment score. Another variable, user visibility or popularity, as measured

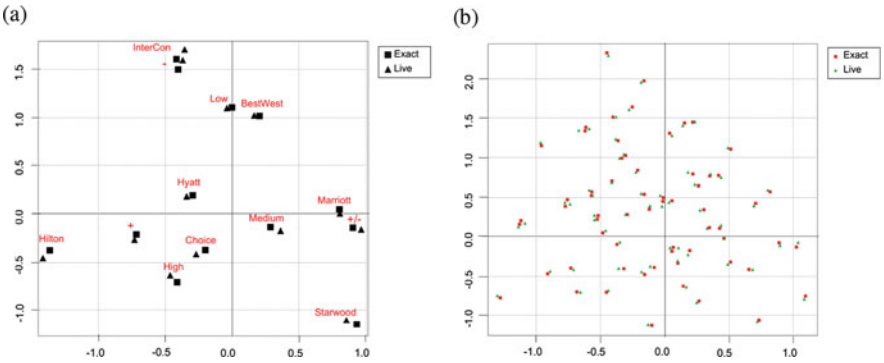

Fig. 1 Exact and Live MCA map of attributes and tweets. (**a**) Attributes. (**b**) Tweets

by the number of followers each user had, was also included in the data set. The variable was categorized into three groups, "low," "medium," and "high."

The potential of the proposed methodology lies in monitoring the competitive position of each brand relative to rival brands over time, as new data blocks are processed, according to standard MCA interpretation. The first block consisted of 500 rows (tweets), and five equally sized blocks were consequently added to update the original solution. In Fig. 1, we plot the final solutions of both approaches (Exact and Live) on the same map for the attributes and tweets, respectively. The similarity between the two configurations was measured by the R index, that equals $\sqrt{1-m^2}$, where m^2 is the symmetric orthogonal Procrustes statistic. The index ranges from 0 to 1 and can be interpreted as a correlation coefficient. The similarity was found very high ($R = 0.997$ for both tweets and attributes), which indicates that the "Live" approach in this case is highly accurate. An implementation in the R language available on http://www.amarkos.gr/research/dynMCA/ allows the reader to directly experiment with the proposed methods.

6 Conclusions

The applicability of GCANO and related SVD-based methods has been extended to big data settings and data streams. The so-called Exact implementation leads to an exact solution and follows the split-apply-combine paradigm; this is a desirable feature for the incremental processing of previously available data or for parallel execution. The "Live" implementation extends the applicability of GCANO in cases when the whole data set is not available from the beginning, as in the case of data streams. The discrepancy between Live and ordinary approaches, as well as the accuracy of the Live approach are thoroughly discussed in Iodice D' Enza and Markos (2015), albeit only in the case of indicator variables. These properties are left to be studied in the case of continuous variables. Since the proposed incremental

approach was based on sequential decomposition, an interesting perspective would be to investigate its relationship with a probabilistic approach, i.e. based on stochastic approximation. We defer consideration of these possibilities to future work.

References

Baglama, J., & Reichel, L. (2007). Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing, 27*, 19–42.

Baker, C., Gallivan, K., & Van Dooren, P. (2012). Low-rank incremental methods for computing dominant singular subspaces. *Linear Algebra and its Applications 436*(8), 2866–2888.

Bijmolt, T. H., & Van de Velden, M. (2012). Multiattribute perceptual mapping with idiosyncratic brand and attribute sets. *Marketing Letters, 23*(3), 585–601.

Carroll, J. D. (1968). A generalization of canonical correlation analysis to three or more sets of variables. In *Proceedings of the 76th Annual Convention of the American Psychological Association* (pp. 227–228).

Correa, N. M., Eichele, T., Adali, T., Li, Y., & Calhoun, V. D. (2010). Multi-set canonical correlation analysis for the fusion of concurrent single trial ERP and functional MRI. *Neuroimage, 50*, 1438–1445.

Gentry, J. (2011). *twitteR: R based twitter client*, http://cran.r-project.org/web/packages/twitteR/

Gifi, A. (1990). *Nonlinear multivariate analysis*. New York: Wiley.

Golub, G., & Van Loan, A. (1996). *Matrix computations*. Baltimore: John Hopkins University Press.

Herbster, M., & Warmuth, M. K. (2001). Tracking the best linear predictor. *Journal of Machine Learning Research, 1*, 281–309.

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

Kroonenberg, P. M. (2008). *Applied multiway data analysis*. New York: Wiley.

Ross, D., Lim, J., Lin, R. S., & Yang, M. H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision, 77*, 125–141.

Takane, Y., Hwang, H., & Abdi, H. (2008). Regularized multiple-set canonical correlation analysis. *Psychometrika, 73*(4), 753–775.

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 61*(3), 611–622.

Van de Velden, M., & Takane, Y. (2012). Generalized canonical correlation analysis with missing values. *Computational Statistics, 27*(3), 551–571.

Van der Burg, E. (1988). *Nonlinear canonical correlation and some related techniques*. Leiden: DSWO Press.

Wickam, H. (2011). A split-apply-combine strategy for data analysis. *Journal of Statistical Software 11*(1), 1–29.