Chemometrics and Intelligent Laboratory Systems 130 (2014) 6–13

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

journal homepage: www.elsevier.com/locate/chemolab

Dimensionality choice in principal components analysis via cross-validatory methods

Peyman Eshghi

Department of Statistical Sciences, GlaxoSmithKline, Park Road, Hertfordshire SG12 0DP, UK

ARTICLE INFO

Article history:
Received 11 June 2013
Received in revised form 30 August 2013
Accepted 5 September 2013
Available online 13 September 2013

Keywords:
NIPALS
PCA
Cross-validation

ABSTRACT

This paper considers cross-validation based approaches to automatically determine the appropriate number of dimensions to retain in a Principal Components Analysis (PCA). Three approaches based on a mixture of leaving groups of observations and variables out are described. They are compared through simulation across a range of datasets of differing sizes and differing levels of missingness using the NIPALS algorithm to carry out the PCA. Also included in the paper is an explicit description of how the NIPALS algorithm is implemented to deal with missing data. Finally we provide suggestions as to which approach offers a better compromise between reliability in choosing the optimal number of components, and the computational burden.

1. Introduction

Principal Component Analysis (PCA) continues to be one of the most commonly used approaches to chemometric data analysis providing insight into (often high-dimensional) multivariate data by a lower dimensional representation. PCA provides components which are a linear combination of the variables where most of the data's variability is usually contained in the first few components. There are a number of mechanisms that are commonly used to determine the optimum number of components. Jackson [1] covers a few of these in great detail in his book and he asserts that cross-validation techniques have evident advantages over most of the existing techniques. The optimum number of components refers to the minimum number of components required to sufficiently explain the data. However, it may be necessary to take into account other factors when deciding the optimum number of components, such as the purpose of the PCA, or the cost of using too many or too few components. The suggestion for using cross-validation to determine the optimum number of principal components was first introduced by Wold [2] and was further developed by Eastment and Krzanowski [3]. Krzanowski and Eastment's approach (K + E) assumed complete data and carried out a full leave-one-out cross-validation using the singular value decomposition for performing the PCA. In practice, PCA is regularly applied to incomplete data via Nonlinear Iterative Partial List Squares (NIPALS) algorithm [4]. In addition concerns over the heavy computational burden, and the risk of over-fitting due to perturbations, both associated with leave-one-out approaches in large datasets [5], have led to a modified version of K + E being implemented in SIMCA-P + © [6]. In 2008, Bro et al. [7] critically looked at some of the existing cross-validation methods used in software packages, highlighting their deficiencies. Here we implement, compare, and evaluate three different variations of cross-validation techniques using the NIPALS algorithm to carry out the PCA. In the sections that follow we provide explicit details of the NIPALS algorithm applied to incomplete data matrices, discuss how to implement the standard, K + E, and Modified K + E cross-validation algorithms and carry out a number of simulation studies in R [8] to investigate their performance for different sizes of data matrix and different levels of missingness. Our conclusion is based on the performance of three methods on simulated datasets defined by six datasets. In turn, these datasets are random samples, taken from two original real-life datasets with different levels of missingness. We understand that this is potentially a major limitation of our study.

2. Methods

2.1. PCA

Background references to PCA are given in numerous papers and books [1,9–11]. We do not intend to add to this but we simply provide the following to introduce the notation and terminology used in this paper.

In what follows we shall use bold uppercase letters (e.g. **X**) to represent matrices, bold lowercase letters (e.g. **x**) to represent vectors and normal lowercase letters (e.g. *x*) to represent scalars.

E-mail address: peyman.5.eshghi@gsk.com.

0169-7439/$ – see front matter
http://dx.doi.org/10.1016/j.chemolab.2013.09.004

Let $\mathbf{X}$ represent an $(n \times m)$ data matrix containing the observed values of m measured variables (columns) on n units of observations (rows) so that

$$\mathbf{X} = \left(x_{i,j}\right)_{i=1,\ldots,n;j=1,\ldots,m}$$

where $x_{i,j}$ represents the value of the jth variable on the ith observation.

In addition, we denote the jth column of $\mathbf{X}$ by $\mathbf{x}_j$. Assuming our PCA is based on $m' \leq \min(n,m)$ components, we denote the set of m' "loading" vectors as the $(m \times m')$ matrix $\mathbf{P}$ with individual columns $\mathbf{p_k}(k = 1,\ldots, m')$ and the associated set of m' "scores" vectors as the $(n \times m')$ matrix $\mathbf{T}$ with individual columns $\mathbf{t_k}(k = 1,\ldots, m')$ such that $\mathbf{T} = \mathbf{XP}$.

Whilst a number of algorithms exist for performing PCA, the NIPALS algorithm has found particular favour within the Chemometrics community in part due to the algorithm's ability to produce results in the presence of missing values within the data matrix. Since previously published versions of the NIPALS algorithm have typically been written assuming that the data matrix is complete, we provide algorithmic details below.

In what follows we will assume that prior to applying the algorithm, the 'raw' data matrix has already been pre-treated, e.g. by subtracting column means and dividing by column sample standard deviations based on the non-missing data for the relevant column.

2.1.1. NIPALS algorithm for incomplete data matrix

1. $\mathbf{\Pi^{(0)}} := \mathbf{X}$
   For $k = 1,\ldots, m'$:
2. Initialise $\mathbf{t}_k$, the vector of scores for the kth principal component, for example using the first column of $\mathbf{\Pi}^{(0)}$.
3. Repeat steps 3.1–3.4 until convergence:

   3.1. Update the loading vector, $\mathbf{P}_k$ for the kth component:

   $$p_{jk} := \frac{\sum_{i \in R_j} \pi_{jk}^{(k-1)} t_{ik}}{\sum_{i \in R_j} t_{ik}^2}$$

   where R_j is the set of row subscripts for the non-missing elements of column j of $\mathbf{X}$.

   3.2. Normalise the loading vector for the kth component:

   $$p_{jk} := \frac{p_{jk}}{\sqrt{\sum_j p_{jk}^2}}$$

   3.3 Update the scores vector, $\mathbf{t}_k$ for the kth component:

   $$t_{ik} := \frac{\sum_{j \in C_i} \pi_{ij}^{(k-1)} p_{jk}}{\sum_{j \in C_i} p_{jk}^2}$$

   where $C_i$is the set of column subscripts for the non-missing elements of row i of $\mathbf{X}$.

   3.4. Compute λ_k, the eigenvalue as below:

   $$\lambda_k := \mathbf{t_k}^T \mathbf{t_k}$$

4. $\mathbf{\Pi^{(k)}} = \mathbf{\Pi^{(k-1)}} - \mathbf{t_k p_k^T}$

The above process can easily be converted to a computer program in any high level programming language.

As seen, the data is projected onto the current estimate of the loading or score vector and in the presence of missing data, the minimum distance projections of the missing values are imputed [12]. This shows how one can efficiently utilise NIPALS for model building in the presence of missing values [12].

It is worth emphasising that the handling of missing data is not a straightforward matter and the success is greatly dependent on the size and nature of the missing items. In general, NIPALS is applicable when the missing values are randomly distributed amongst the data, i.e. missing completely at random, and when the number of missing values does not exceed 20% [12,13].

2.2. Choice of number of components

As seen, NIPALS extracts one principal component at a time and it can stop when the desired number of components is retained but there are no universally accepted rules for making a decision on the sufficient number of principal components. However there are a number of mechanisms that are commonly used to determine the optimum number of components; range from significance test to graphical procedures. Velicer's Partial Correlation Procedure [14], The Scree Test [15], and Malinowski's Indicator function [16] are just a few of these techniques that introduce criteria for optimality. Amongst different methods, Jackson [1] asserts that cross-validation techniques have evident advantages over most of the existing techniques. Two commonly used cross-validation methods, together with a previously unpublished technique will be explained. Our aim is to explain these methods with a sufficient level of algorithmic details so that with a comprehensive simulation study we can assess the weakness and strength of the methods, both statistically and computationally.

2.3. Standard cross-validation

Standard cross-validation in PCA, as describe by Jackson [1], starts with randomly subdividing the data matrix into 'g' groups of n/g observations each. Without loss of generality we order the subgroups as they appear in the original data matrix $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \hline \mathbf{X}_2 \\ \hline \vdots \\ \hline \mathbf{X}_g \end{bmatrix}$$

Each subgroup will then be left out from the dataset in turn and NIPALS is applied on the remaining data to obtain the first k loading vectors for them:

$$(\mathbf{P} = \boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_k}) \leftarrow NIPALS\left(\left(\mathbf{X}^{(-t)}\right), m' = k\right) \quad (1)$$

$\mathbf{X}^{(-t)}$ denotes the remaining data after leaving tth subgroup out (i.e. $[\mathbf{X}_1^T,\ldots,\mathbf{X}_{t-1}^T, \mathbf{X}_{t+1}^T,\ldots,\mathbf{X}_g^T]^T$) where $t = 1,2,\ldots,g$

We can estimate the score matrix $\mathbf{T}$ for the left out subgroup $\mathbf{X}_t$:

$$\mathbf{T} = \mathbf{X}_t \mathbf{P} \quad (2)$$

We can now estimate values for the left out subgroup based on the first k components:.

$$\hat{\mathbf{X}}_t^{(k)} = \mathbf{T}\ \mathbf{P}^T \quad (3)$$

$\hat{\mathbf{X}}_t^{(k)}$ denotes the matrix of estimated values for the left out subgroup $\mathbf{X}_t$ based on the first k components.

Up to $m - 1$ sets of predicted values will be obtainable for each left out group via this method by separately using the first component, first two components, first three components, and so on:

$$\hat{\mathbf{X}}_i^{(1)}, \hat{\mathbf{X}}_i^{(2)}, \ldots, \hat{\mathbf{X}}_i^{(m')} \text{ where } m' < m$$