Forward Selection Component Analysis: Algorithms and Applications

Puggini, L., & McLoone, S. (2017). Forward Selection Component Analysis: Algorithms and Applications. IEEE Transactions on Pattern Analysis and Machine Intelligence. DOI: 10.1109/TPAMI.2017.2648792

Published in:
IEEE Transactions on Pattern Analysis and Machine Intelligence

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Forward Selection Component Analysis: Algorithms and Applications

Luca Puggini, Seán McLoone, *Senior Member, IEEE*

Abstract—Principal Component Analysis (PCA) is a powerful and widely used tool for dimensionality reduction. However, the principal components generated are linear combinations of all the original variables and this often makes interpreting results and root-cause analysis difficult. Forward Selection Component Analysis (FSCA) is a recent technique that overcomes this difficulty by performing variable selection and dimensionality reduction at the same time. This paper provides, for the first time, a detailed presentation of the FSCA algorithm, and introduces a number of new variants of FSCA that incorporate a refinement step to improve performance. We then show different applications of FSCA and compare the performance of the different variants with PCA and Sparse PCA. The results demonstrate the efficacy of FSCA as a low information loss dimensionality reduction and variable selection technique and the improved performance achievable through the inclusion of a refinement step.

Index Terms—Unsupervised dimensionality reduction, subset selection, feature selection

1 Introduction

The need to analyse large volumes of multivariate data is an increasingly common occurrence in many areas of science, engineering and business. In order to build more interpretable models or to reduce the cost of data collection it is important to discover good compact representations of high-dimensional datasets. This leads to the fundamental problem of dimensionality reduction. Many methods have been developed to perform supervised dimensionality reduction [1], [2], [3], [4] . Given an input matrix $\mathbf{X} \in \mathbb{R}^{m \times v}$ (containing m measurements of v variables) and an output value $\mathbf{y} \in \mathbb{R}^m$ these methods try to understand what subset of variables, or derived features of $\mathbf{X}$ optimally explain $\mathbf{y}$. Dimensionality reduction can also be defined as an unsupervised problem. In this case we look for the subset of variables/derived features that retain the maximum information content with respect to the original set of variables, in the sense of being able to reconstruct the full data matrix $\mathbf{X}$. Different unsupervised dimensionality reduction techniques have been proposed. Some of them, such as [5], [6], [7], [8], have been developed with the goal of maximising performance when used as a pre-processing step in clustering or classification algorithms, while others, such as [9], [10], [11], [12], have been developed in order to obtain the optimal reconstruction of the full dataset. Among this latter group Principal Component Analysis (PCA) is the best known and most widely used technique [11]. PCA provides the most efficient linear transformation of data to a lower dimensional space and is relatively straight forward to compute. It has found many applications in chemometrics and other fields where datasets are encountered involving large numbers of variables with significant levels of inter-variable correlation and hence redundancy (see for example [13] and [14]). However, while PCA provides a compact representation of a multivariate dataset, it does not lend itself to identification of the most representative subset of variables within the data. This is a consequence of the fact that the latent variables (principal components) generated by PCA are a linear combination of all original variables, making the most significant variables difficult to determine [15]. This is especially true in the case of highly correlated datasets due to the grouping effect, whereby the contribution of a group of highly correlated variables to a given principal component is distributed evenly across all variables in the group. While this characteristic is beneficial in terms of noise suppression, it means that the contribution of individual variables can be small making important variables appear insignificant. Hence, tasks such as identification of key variables, root-cause analysis and model interpretation can be challenging using PCA.

Consequently, various approaches have been developed to obtain sparse approximations of PCA. The simplest strategy is to manually set to 0 the values of the principal components (PCs) that are smaller than a given threshold but this can lead to significant variables being missed if they are part of a group of highly correlated variables [16], [15]. More sophisticated approaches such as SCoTLASS [17], DSPCA [18], sparse PCA [19], sPCA-rSVD [20], SOCA [21] and [22] use a lasso like L_1 or L_0 penalty or are formulated as constrained maximization problems in order to encourage sparsity in the PCA loadings. However, these methods are generally computationally intensive and difficult to use and interpret due to the need to establish the appropriate level of sparsity for each PC computed.

These challenges motivated the second author to develop a technique called Forward Selection Component Analysis (FSCA) which seeks to identify a small number

- *L. Puggini is with the Department of Electronic Engineering, National University of Ireland, Maynooth. E-mail: luca.puggini@gmail.com, luca.puggini.2014@mumail.ie*
- *S. McLoone is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast. E-mail: s.mcloone@qub.ac.uk*

Manuscript received June 15, 2015; revised May 23, 2016; accepted December 30, 2016.

of key variables that are representative of the observed variance across all variables. FSCA was initially introduced in the context of Optical Emission Spectroscopy (OES) data analysis of plasma etch processes [23] where isolating a small number of wavelengths is important for understanding the underlying plasma chemistry. More recently, FSCA has been found to be a particularly effective tool for optimising measurement site selection for spatial wafer metrology in semiconductor manufacturing [24]. The method works by iteratively deriving a set of orthogonal components which are a function of only a subset of the original variables, and which sequentially maximize the explained variance. At one level FSCA can be regarded as the unsupervised counterpart of Forward Selection Regression in that it returns a set of Forward Selected Variables (FSVs), but equally it retains some of the characteristics and utility of PCA in that it also returns a set of Forward Selection Components (FSCs) which form an orthogonal basis. This allows, for example, the contributions of individual components to be easily isolated.

In this paper, we present for the first time, a complete description of the FSCA procedure and algorithms, drawing parallels with PCA as appropriate. In addition, motivated by the success of a two stage algorithm proposed in [25], [26] for stepwise regression based methods, we propose a number of new variants of FSCA that incorporate a similar backward refinement step. Specifically, we introduce four backward refinement variants, which we call Single, and Multi-pass Backward Refinement FSCA (denoted as SPBR-FSCA and MPBR-FSCA); and Recursive Single, and Recursive Multi-pass Backward Refinement FSCA (denoted as R-SPBR-FSCA and R-MPBR-FSCA). Then, with the aid of a number of case studies, we demonstrate the utility of FSCA, the enhanced performance obtained with the new variants, and provide comparisons with PCA and Sparse PCA.

The remainder of the paper is organised as follows. Related work on variable selection techniques is identified in Section 2, followed by the algorithmic descriptions of PCA and FSCA in Section 3. Section 4 introduces the new backward refinement FSCA algorithms. Then comparative results and analysis are provided in Section 5 and 6 for simulated and real world case studies, respectively. Finally, conclusions are presented in Section 7.

2 Related Work

A variety of variable selection methods have been developed based on making comparisons with or extracting information from a PCA decomposition of the data matrix (e.g. [27], [28], [29], [30] and [31]). Other approaches employ clustering of features using a suitable feature similarity metric as the basis for variable selection (e.g. [5], [7] and [12]). Recently [32] proposed a novel L_1 regularised formulation for the unsupervised variable selection problem which has a similar philosophy to sparse PCA and can be thought of as the unsupervised counterpart of LASSO [1]. In addition to FSCA, two other techniques which can be considered as performing direct variable selection are the algorithms by Whitley et al. [33] and Wei and Billings [34], both of which employ orthogonalisation procedures. In the former, variables are selected based on sequentially finding the variables in the dataset that are most uncorrelated with linear combinations of the variables already selected, while in the latter the criterion used for variable selection is the maximum average squared correlation with all other variables in the dataset. Wei and Billings's algorithm, which they refer to as Forward Orthogonal Search (FOS), is similar in character to FSCA and, as will be discussed in Section 3.2, yields identical results to FSCA if the data is appropriately pre-scaled. Recently [35] introduced a kernel extension of variable selection that enables non-linear relationships between variables to be taken in account, while [36] developed an efficient parallel implementation for data parallel distributed computing that scales well for large problems. Both these algorithms are equivalent to FSCA in terms of the sequence of variables selected, but operate directly in the variable space rather than producing FSCs.

While FSCA and the proposed backward refinement variants are specifically designed for unsupervised variable selection, we note that they have a number of characteristics in common with more general machine learning and signal processing dictionary selection, sparse representation and supervised variable selection problems with regard to the use of greedy forward selection and backward refinement steps. In [37], for example, a supervised dictionary selection framework is developed for spare representation of a set of signals where the dictionary elements are recursively selected from a candidate set $\mathcal{D}$ using a greedy forward selection algorithm. FSCA can be regarded as special case of this framework, where the signals to be represented are also the set of candidate dictionary elements $\mathcal{D}$.

In the supervised context, in particular, various backward refinement algorithms have been proposed to address the non-optimality of greedy selection. Two notable examples are the FoBa algorithm by Zhang [38] for learning sparse representations and the Orthogonal Matching Pursuit with Replacement (OMPR) algorithm by Jain *et al.* [39] for compressed sensing. In FoBa following each greedy selection step a backward variable elimination step is performed to remove variables that are not contributing to the model. The backward step is aggressive in that it allows a small increase in error when a variable is removed. This is limited to be half the error reduction in the corresponding forward selection step such that algorithm convergence is guaranteed. In OMPR an initial set of variables k is randomly selected and then a refinement step is repeatedly performed in which variables are replaced with new ones from the candidate set using a gradient based update procedure.

The backward refinement FSCA algorithms presented in this paper share some similarities with both FoBa and OMPR. FoBa is essentially a recursive multi-pass backward refinement procedure, but differs from R-MPBR-FSCA in that variables are removed rather than replaced in the refinement step. In addition, while our refinement algorithms are strictly hill climbing, i.e. variables are only replaced if they lead to an improvement in performance, they can easily be adapted to employ the more aggressive backward refinement step of FoBa. Both OMPR and backward refinement FSCA employ a variable replacement strategy, and while OMPR differs substantially from FSCA algorithmically, it can be regarded as a multi-pass backward refinement approach, but with the k components initialized randomly,

rather than being obtained as the output of a forward selection procedure.

More generally, convex relaxations of the variable selection problem are a particular case of optimization over convex hulls of an atomic set [40], which can be effectively solved using greedy Frank-Wolfe (aka conditional gradient) type algorithms [41]. In this context [42] have developed an optimization procedure called CoGEnT for general atomic-norm regularization problems which incorporates both a greedy forward selection step and an aggressive backward refinement step and is effectively a generalization of FoBa to atomic norms.

3 Data Decomposition and Reconstruction

Given a matrix $\mathbf{X} \in \mathbb{R}^{m \times v}$ representing a dataset with m measurements of v variables, where each variable can be considered without loss of generality to be normalised to have zero mean, different techniques can be used to obtain a more compact representation of $\mathbf{X}$. Our aim is to estimate a matrix $\mathbf{S} \in \mathbb{R}^{m \times k}$, where $k < rank(\mathbf{X}) \leq min(m, v)$, such that it is possible to obtain a good reconstruction of $\mathbf{X}$ by linear regression on $\mathbf{S}$:

$$\hat{\mathbf{X}} = \mathbf{S}\mathbf{\Theta}. \tag{1}$$

In general, given a regressor matrix $\mathbf{S}$, the optimal least square error linear reconstruction of the original signal $\mathbf{X}$ is given by

$$\mathbf{\Theta} = \underset{\tilde{\mathbf{\Theta}}}{argmin} \| \mathbf{S}\tilde{\mathbf{\Theta}} - \mathbf{X} \|_F^2, \tag{2}$$

where $(\| \cdot \|_F)$ is the Frobenius norm. The solution to (2) is the well known least-squares solution:

$$\mathbf{\Theta} = (\mathbf{S}^\mathsf{T}\mathbf{S})^{-1}\mathbf{S}^\mathsf{T}\mathbf{X}. \tag{3}$$

Hence, defining the projection matrix

$$\mathbf{\Phi}(\mathbf{S}) = \mathbf{S}(\mathbf{S}^\mathsf{T}\mathbf{S})^{-1}\mathbf{S}^\mathsf{T}, \tag{4}$$

for a given matrix $\mathbf{S}$, the optimal linear reconstruction of $\mathbf{X}$ can be expressed as

$$\hat{\mathbf{X}} = \mathbf{\Phi}(\mathbf{S})\mathbf{X}. \tag{5}$$

Different metrics can be used to quantify the approximation error between the matrix $\mathbf{X}$ and its reconstruction $\hat{\mathbf{X}}$ (such as element-wise or induced L_1, L_2 and L_∞ norms of $\hat{\mathbf{X}} - \mathbf{X}$). The metric that is often considered when working with Principal Component Analysis, and the one adopted here, is the percentage of explained variance, that is, the percentage of the variance observed in $\mathbf{X}$ explained by $\hat{\mathbf{X}}$. Noting that the columns of $\mathbf{X}$ have been defined as having zero mean, this can be expressed in terms of the Frobenius norm as

$$V_{\mathbf{X}}(\hat{\mathbf{X}}) = 100 \times \left(1 - \frac{\| \hat{\mathbf{X}} - \mathbf{X} \|_F^2}{\| \mathbf{X} \|_F^2}\right). \tag{6}$$

While $V_{\mathbf{X}}(\hat{\mathbf{X}})$ is unbounded in the negative direction for arbitrary $\hat{\mathbf{X}}$, when $\hat{\mathbf{X}}$ is computed as a projection of $\mathbf{X}$ onto the subspace spanned by $\mathbf{S}$, as given by eqt. (5), $V_{\mathbf{X}}(\hat{\mathbf{X}}) \geq 0$ for arbitrary $\mathbf{S}$. (A proof is provided in Appendix A.)

3.1 Principal Component Analysis

Principal Component Analysis (PCA) is one of the most common and widely used dimensionality reduction techniques. The PCA decomposition of $\mathbf{X}$ is defined as

$$\hat{\mathbf{X}}_{PCA}^k = \mathbf{T}_k\mathbf{P}_k^\mathsf{T} = \sum_{i=1}^{k} \mathbf{t}_i\mathbf{p}_i^\mathsf{T}, \tag{7}$$

where $\mathbf{P}_k \in \mathbb{R}^{v \times k}$ is an orthonormal matrix, computed as the first k ordered eigenvectors of the data covariance matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ (in descending eigenvalue order), and $\mathbf{T}_k \in \mathbb{R}^{m \times k}$ is the geometric projection of $\mathbf{X}$ on the columns of $\mathbf{P}_k$, that is:

$$\mathbf{T}_k = \mathbf{X}\mathbf{P}_k. \tag{8}$$

Here, $\mathbf{p}_i$ and $\mathbf{t}_i$ are the i-th column's of $\mathbf{P}_k$ and $\mathbf{T}_k$, respectively. If $k = r = rank(\mathbf{X})$ then the PCA decomposition is exact and

$$\mathbf{X} = \hat{\mathbf{X}}_{PCA}^r = \mathbf{T}_r\mathbf{P}_r^\mathsf{T}. \tag{9}$$

Otherwise, if $k < rank(\mathbf{X})$ PCA provides the best rank k approximation to $\mathbf{X}$ [11], that is, $\mathbf{P}_k$ is the solution to the optimisation problem

$$\underset{\mathbf{P}_k}{argmax}\ V_{\mathbf{X}}(\mathbf{X}\mathbf{P}_k\mathbf{P}_k^\mathsf{T}). \tag{10}$$

Consequently, it follows that when $\mathbf{S}$ is restricted to k columns, the optimal choice is $\mathbf{S} = \mathbf{T}_k$, in which case $\mathbf{\Theta} = \mathbf{P}_k^\mathsf{T}$.

Various algorithms exist for computing the PCA decomposition. Among these the most popular are the singular value decomposition (SVD) and the Nonlinear Iterative Partial Least Squares (NIPALS) [43] algorithms. While SVD is more numerically robust and efficient when a full PCA decomposition is required, the advantage of NIPALS is that it computes the PCA decomposition iteratively, one principal component (PC) at a time, in descending order. This makes it highly efficient in high dimensional problems where typically only a small number of PCs need to be computed. For completeness, and to facilitate comparison with FSCA presented in the next section, a description of the NIPALS algorithm is provided in Algorithm 1.

3.2 Foward Selection Component Analysis

In contrast to PCA which produces a reduced set of new variables (PCs) that are linear combinations of all existing variables, Forward Selection Component Analysis (FSCA) derives a set of new variables (FSCs) that are a function of only a subset of the original variables that maximise the explained variance. This is achieved using the iterative procedure detailed in Algorithm 2. The FSCA algorithm returns:

- A matrix $\mathbf{Z}_k$ composed of a subset of the columns of $\mathbf{X}$ (FSVs) ranked according to how well they contribute to the reconstruction of $\mathbf{X}$.
- A matrix of FSCA components (FSCs) $\mathbf{M}_k$. The first column of $\mathbf{M}_k$ is equivalent to the first column of $\mathbf{Z}_k$. The second column of $\mathbf{M}_k$ is a function of the first and the second column of $\mathbf{Z}_k$ and so on. In general the k-th FSC will be defined as a function of itself and of the previous $k-1$ components and so is a function of the first k selected variables.

Algorithm 1: $[\mathbf{P_k}, \mathbf{T_k}]$=NIPALS($\mathbf{X}$, k)

Require: Data matrix $\mathbf{X}$, number of PCs k
1: Set $\mathbf{R} = \mathbf{X}$
2: Initialise $\mathbf{P}_0 = \mathbf{T}_0 = \emptyset$
3: Set $\epsilon = 10^{-6}$ (convergence threshold)
4: Initialise $\mathbf{t}$ to a non-zero column of $\mathbf{X}$
5: **for** $j = 1$ **to** k **do**
6: Set $\mathbf{t}_{new} = \mathbf{t}$ and $\mathbf{t}_{old} = 10\mathbf{t}$
7: **while** $\parallel \mathbf{t}_{old} - \mathbf{t}_{new} \parallel_2 \ \geq \ \epsilon$ **do**
8: $\mathbf{t}_{old} = \mathbf{t}_{new}$
9: $\mathbf{p} = \mathbf{R}^\mathsf{T}\mathbf{t}/\mathbf{t}^\mathsf{T}\mathbf{t}$
10: $\mathbf{p} = \mathbf{p}/\sqrt{\mathbf{p}^\mathsf{T}\mathbf{p}}$
11: $\mathbf{t} = \mathbf{R}\mathbf{p}$
12: $\mathbf{t}_{new} = \mathbf{t}$
13: **end while**
14: $\mathbf{P}_j = [\mathbf{P}_{j-1}\ \mathbf{p}]$
15: $\mathbf{T}_j = [\mathbf{T}_{j-1}\ \mathbf{t}]$
16: $\mathbf{R} = \mathbf{R} - \mathbf{t}\mathbf{p}^\mathsf{T}$
17: **end for**
18: **return** $\mathbf{P}_k,\ \mathbf{T}_k$

Fig. 1. PCA NIPALS Algorithm

- A matrix of FSCA loadings $\mathbf{U}_k$.

FSCA leads to a decomposition of $\mathbf{X}$ of the form

$$\hat{\mathbf{X}}^k_{FSC} = \mathbf{M}_k\mathbf{U}_k^\mathsf{T} = \sum_{i=1}^{k} \mathbf{m}_i\mathbf{u}_i^\mathsf{T}. \tag{11}$$

where $\mathbf{M}_k$ is an orthogonal matrix of Forward Selection Components (FSCs). Equivalently we can express the decomposition directly in terms of the Forward Selection Variables (FSVs), $\mathbf{Z}_k$ as

$$\hat{\mathbf{X}}^k_{FSV} = \mathbf{Z}_k\mathbf{B}_k^\mathsf{T} = \sum_{i=1}^{k} \mathbf{z}_i\mathbf{b}_i^\mathsf{T}. \tag{12}$$

Here $\mathbf{Z}_k = [\mathbf{z}_1, ..., \mathbf{z}_k] = [\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}] \subset \mathbf{X}$ and $\mathbf{B}_k^\mathsf{T}$ are the corresponding least squares regression coefficients, that is

$$\mathbf{B}_k = \mathbf{X}^\mathsf{T}\mathbf{Z}_k(\mathbf{Z}_k^\mathsf{T}\mathbf{Z}_k)^{-1}. \tag{13}$$

In a similar fashion to NIPALS, the FSCs generated by FSCA are ordered in descending order in terms of the variance of $\mathbf{X}$ explained. Furthermore, by virtue of their orthogonality the variance contribution of the i-th FSC is simply obtained as $(\mathbf{m}_i^\mathsf{T}\mathbf{m}_i)(\mathbf{u}_i^\mathsf{T}\mathbf{u}_i)$. A similar expression holds for PCA, but since $\mathbf{P}_k$ is an orthonormal matrix this reduces to $\mathbf{t}_i^\mathsf{T}\mathbf{t}_i$.

The FSV decomposition could be computed directly, rather than as a by-product of the FSC computation by solving

$$i_k = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}([\mathbf{Z}_{k-1}\mathbf{x}_i])\mathbf{X}) \tag{14}$$

and setting $\mathbf{Z}_k = [\mathbf{Z}_{k-1}\ \mathbf{x}_{i_k}]$, with $\mathbf{Z}_0 = \emptyset$. However, the results are equivalent for a given number of components, that is $\hat{\mathbf{X}}^k_{FSC} = \hat{\mathbf{X}}^k_{FSV}$. In particular, once $\mathbf{Z}_k$ has been computed, the corresponding FSC matrix $\mathbf{M}_k$ can be obtained as the Gram-Schmidt orthogonalization of $\mathbf{Z}_k$. Thus an

Algorithm 2: $[\mathbf{Z}_k, \mathbf{M}_k, \mathbf{U}_k]$=FSCA($\mathbf{X}$, k)

Require: Data matrix $\mathbf{X}$, number of FSCs k
1: Set $\mathbf{R} = \mathbf{X}$ # Notation: $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_p]$
2: Initialise $\mathbf{Z}_0 = \mathbf{M}_0 = \mathbf{U}_0 = \emptyset$
3: **for** $j = 1$ **to** k **do**
4: $i = \underset{\mathbf{r}_i \in \mathbf{R}}{argmax}\ V_{\mathbf{R}}(\mathbf{\Phi}(\mathbf{r}_i)\mathbf{R})$
5: $\mathbf{m} = \mathbf{r}_i$
6: $\mathbf{z} = \mathbf{x}_i$
7: $\mathbf{u} = \mathbf{R}^\mathsf{T}\mathbf{r}_i/(\mathbf{r}_i^\mathsf{T}\mathbf{r}_i)$
8: $\mathbf{M}_j = [\mathbf{M}_{j-1}\ \mathbf{m}]$
9: $\mathbf{Z}_j = [\mathbf{Z}_{j-1}\ \mathbf{z}]$
10: $\mathbf{U}_j = [\mathbf{U}_{j-1}\ \mathbf{u}]$
11: $\mathbf{R} = \mathbf{R} - \mathbf{\Phi}(\mathbf{m})\mathbf{R}$
12: **end for**
13: **return** $\mathbf{Z}_k,\ \mathbf{M}_k,\ \mathbf{U}_k$

Fig. 2. FSCA Algorithm

Algorithm 3: $[\mathbf{Z_k}, \mathbf{M_k}, \mathbf{U_k}]$=FSVA($\mathbf{X}$,$k$)

Require: Data matrix $\mathbf{X}$, number of FSVs k
1: $\mathbf{Z}_0 = \emptyset$
2: **for** $j = 1$ **to** k **do**
3: $i_j = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}([\mathbf{Z}_{j-1}\ \mathbf{x}_i])\mathbf{X})$
4: $\mathbf{Z}_j = [\mathbf{Z}_{j-1}\ \mathbf{x}_{i_j}]$
5: *Optional refinement step (recursive)*
6: **end for**
7: *Optional refinement step*
8: $\mathbf{M}_k = \text{GramSchmidt}(\mathbf{Z}_k)$
9: $\mathbf{U}_k = \mathbf{X}^\mathsf{T}\mathbf{M}_k(\mathbf{M}_k^\mathsf{T}\mathbf{M}_k)^{-1}$
10: **return** $\mathbf{Z}_k,\ \mathbf{M}_k,\ \mathbf{U}_k$

Fig. 3. Direct FSV implementation of FSCA

alternative FSV based implementation of FSCA is as given in Algorithm 3. Note that steps 5 and 7 are place holders for the refinement step which will be introduced in Section 4 and are not part of the basic algorithm, which we will refer to as the FSV algorithm (FSVA). If we are only interested in FSVs steps 8 and 9 can be omitted, in which case the algorithm corresponds to the feature selection methods presented in [35] and [36].

Since PCA provides the optimal representation in terms maximising the variance explained (eq. 10) it follows that $V_{\mathbf{X}}(\hat{\mathbf{X}}^k_{FSC}) \leq V_{\mathbf{X}}(\hat{\mathbf{X}}^k_{PCA})$. Hence PCA can be regarded as providing an upper bound on the performance achievable with FSCA with a given number of variables k, or equivalently a lower bound on the number of variables needed to achieve a desired reconstruction accuracy.

3.3 Computational Complexity of FSCA

The computation time of the FSCA algorithm (Algorithm 2) is dominated by the combinatorial optimisation problem in step 4. It is relatively straight forward to show that maximis-

ing the explained variance is equivalent to maximising the Rayleigh Quotient of $\mathbf{X}\mathbf{X}^\mathsf{T}$ (see Appendix A), that is

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{x}_i)\mathbf{X}) \equiv \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ \frac{\mathbf{x}_i^\mathsf{T}\mathbf{X}\mathbf{X}^\mathsf{T}\mathbf{x}_i}{\mathbf{x}_i^\mathsf{T}\mathbf{x}_i}, \tag{15}$$

which can be computed efficiently as

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{v} \frac{(\mathbf{x}_i^\mathsf{T}\mathbf{x}_j)^2}{\mathbf{x}_i^\mathsf{T}\mathbf{x}_i}. \tag{16}$$

It is interesting to note that the corresponding expression for maximising the average squared correlation metric employed in the FOS algorithm proposed by Wei and Billing [34] is

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{v} \frac{(\mathbf{x}_i^\mathsf{T}\mathbf{x}_j)^2}{(\mathbf{x}_i^\mathsf{T}\mathbf{x}_i)(\mathbf{x}_j^\mathsf{T}\mathbf{x}_j)}. \tag{17}$$

Hence, while the FOS optimization objective has an additional scaling factor in the denominator, both algorithms will in fact yield identical results provided the columns of the data matrix $\mathbf{X}$ are normalised so that they are all the same length ($\mathbf{x}_j^\mathsf{T}\mathbf{x}_j$ will then be invariant with respect to j).

It is also worth noting that if $\mathbf{x}_i$ is not constrained to be a column of $\mathbf{X}$ the solution to (15) is the largest eigenvector of $\mathbf{X}\mathbf{X}^\mathsf{T}$, but this is simply the direction of the score vector $\mathbf{t}_1$ corresponding to the first PC of the data. This suggests an alternative variable selection approach, as proposed by Cui and Dy [31], whereby the first variable selected is the one that is most closely correlated with the first PC. In subsequent steps the selected variables are the closest to the first PC of the corresponding residual matrix. The approach, referred to as Orthogonal Principal Feature Selection (OPFS), is in general only an approximation to FSCA. This can be deduced as follows. Recalling the definition of the PCA decomposition in equations (7)-(9), the FSCA optimization objective (eqt. 15) can be expressed as

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{r} \frac{(\mathbf{x}_i^\mathsf{T}\mathbf{t}_j)^2}{\mathbf{x}_i^\mathsf{T}\mathbf{x}_i} \tag{18}$$

or equivalently as

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{r} \lambda_j corr(\mathbf{x}_i, \mathbf{t}_j)^2, \tag{19}$$

where $\lambda_j (= \mathbf{t}_j^\mathsf{T}\mathbf{t}_j)$ is the variance contribution of the j-th PC. In contrast the OPFS optimization objective corresponds to

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ corr(\mathbf{x}_i, \mathbf{t}_1)^2. \tag{20}$$

Thus, while OPFS selects variables based on their squared correlation with the first PC, FSCA selects them based on the variance-weighted average squared correlation with all PCs. Hence, the sequence of variables selected by OPFS will in general differ from, and explain less variance than, the variables selected by FSCA.

If FSCA is computed using the FSVA implementation (Algorithm 3) an efficient solution can be obtained by noting that the combinatorial optimisation problem in equation (14) is equivalent to

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{v} \mathbf{x}_j^\mathsf{T}\mathbf{\Phi}([\mathbf{Z}_{k-1}\ \mathbf{x}_i])\mathbf{x}_j. \tag{21}$$

Recalling the definition of $\mathbf{\Phi}$ (eqt. 4) this can be recast as

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{v} \mathbf{q}_{j(i)}^\mathsf{T}(\mathbf{Z}_{(i)}^\mathsf{T}\mathbf{Z}_{(i)})^{-1}\mathbf{q}_{j(i)}, \tag{22}$$

where $\mathbf{q}_{j(i)} = \mathbf{Z}_{(i)}^\mathsf{T}\mathbf{x}_j$, and $\mathbf{Z}_{(i)} = [\mathbf{Z}_{k-1}\ \mathbf{x}_i]$. Hence, determining the optimum $\mathbf{x}_i$ requires v^2 evaluations of the vector terms $\mathbf{q}_{j(i)}$ and v evaluations of the matrix inverse term $(\mathbf{Z}_{(i)}^\mathsf{T}\mathbf{Z}_{(i)})^{-1}$. We can take two steps to substantially reduce the computation time for these terms. Firstly, as proposed in [35], [36], an $\mathsf{O}(k^2)$ complexity recursive computation of the matrix inverse can be obtained by taking advantage of the fact that

$$\mathbf{Z}_{(i)}^\mathsf{T}\mathbf{Z}_{(i)} = \begin{bmatrix} \mathbf{Z}_{k-1}^\mathsf{T}\mathbf{Z}_{k-1} & \mathbf{r}_{(i)} \\ \mathbf{r}_{(i)}^\mathsf{T} & a_{(i)} \end{bmatrix}, \tag{23}$$

where $\mathbf{r}_{(i)} = \mathbf{Z}_{k-1}\mathbf{x}_i$, and $a_{(i)} = \mathbf{x}_i^\mathsf{T}\mathbf{x}_i$, and applying block matrix inversion algebra to obtain an expression for $(\mathbf{Z}_{(i)}^\mathsf{T}\mathbf{Z}_{(i)})^{-1}$ in terms of $(\mathbf{Z}_{k-1}^\mathsf{T}\mathbf{Z}_{k-1})^{-1}$ which has already been computed in the previous iteration, that is:

$$(\mathbf{Z}_{(i)}^\mathsf{T}\mathbf{Z}_{(i)})^{-1} = \begin{bmatrix} (\mathbf{Z}_{k-1}^\mathsf{T}\mathbf{Z}_{k-1})^{-1} + w\mathbf{b}\mathbf{b}^T & -w\mathbf{b} \\ -w\mathbf{b}^T & w \end{bmatrix}, \tag{24}$$

$$\mathbf{b} = (\mathbf{Z}_{k-1}^T\mathbf{Z}_{k-1})^{-1}\mathbf{r}_{(i)},\ w = (a_{(i)} - \mathbf{r}_{(i)}^T\mathbf{b})^{-1}.$$

In contrast direct calculation of the matrix inverse has $\mathsf{O}(k^3)$ computational complexity.

Secondly, evaluating the terms $\mathbf{q}_{j(i)}$, $\mathbf{r}_{(i)}$ and $a_{(i)}$ all involve computing vector products $\mathbf{x}_i^\mathsf{T}\mathbf{x}_j$ many times, with substantial repetition both within each variable selection iteration and between iterations. This repetition can be eliminated by precomputing the covariance matrix $\mathbf{C} = \mathbf{X}^\mathsf{T}\mathbf{X}$, where $c_{ij} = \mathbf{x}_i^\mathsf{T}\mathbf{x}_j$, at the cost of $\mathsf{O}(v^2m)$ floating point operations (flops) and $\mathsf{O}(v^2)$ additional memory.

Table 1 shows the estimated complexity in terms of floating point operations for computing k FSCs with the FSCA and FSVA algorithms, with and without the covariance matrix precomputed, while Fig. 4 shows how complexity varies as a function of v and m for specific combinations of the other dimensions. As can be seen, the reduction in complexity is of the order $\mathsf{O}(k^2)$ for FSVA. Precomputing $\mathbf{C}$ is also beneficial for FSCA, but the impact is less significant since it has to be re-computed at each iteration due to the deflation step. That said, a factor of two reduction in computational complexity is achieved for FSCA. All algorithms scale quadratically with v and linearly with m, but differ in how they behave with respect to k, with FSCA implementations growing linearly and FSVA implementations growing cubically. If precomputing the covariance matrix is not an issue, the preferred algorithm is FSVA when $k < \sqrt{1.5m}$ (approx.) and FSCA otherwise. FSCA is substantially superior to FSVA when the covariance matrix is not precomputed and also outperforms precomputed FSVA when $k > \sqrt{3m}$.

When the computational burden of FSCA becomes prohibitive OPFS may offer an attractive compromise due to its significantly lower computational complexity. In OPFS the PC needed at each step can be computed efficiently using NIPALS yielding and algorithm with $\mathsf{O}((4\alpha + 8)mv)$ complexity per selected variable, compared to order $\mathsf{O}(mv^2)$ with FSCA. Here, α denotes the average number of iterations per selected variable for the NIPALS algorithm to

TABLE 1
Flop count and asymptotic complexity for computing k FSCs of $\mathbf{X} \in \mathbb{R}^{m \times v}$ with different FSCA algorithm implementations

Method	Floating point operation count	Complexity $k << v$
FSCA	$\mathsf{O}(2v^2mk + 6vmk + v^2k + 2mk - vk - k)$	$\mathsf{O}(2v^2mk)$
FSCA (PC)	$\mathsf{O}(v^2mk + 2vmk + 2v^2k)$	$\mathsf{O}(v^2mk)$
FSVA	$\mathsf{O}(2v^2m\bar{k}+2v^2\bar{k}^2+4vm\bar{k}+4v\bar{k}^2+4vmk-2vk+2mk^2+4mk)$	$\mathsf{O}(v^2mk^2+\frac{2}{3}v^2k^3)$
FSVA (PC)	$\mathsf{O}(v^2m + v^2(2\bar{k}^2 + \bar{k}) + 2mk^2 + 2vmk+4mk-vk+v(4\bar{k}^2-6\bar{k}+3k))$	$\mathsf{O}(v^2m+\frac{2}{3}v^2k^3)$

Notation: $\bar{k} = k(k-1)/2$; $\bar{k}^2 = k(k+1)(2k+1)/6$; and PC denotes implementation with precomputed covariance matrix.

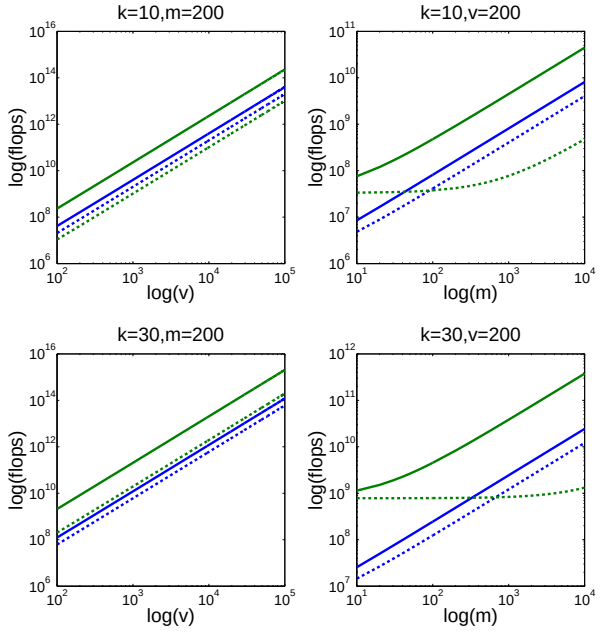

Fig. 4. Complexity of FSCA algorithms as a function of v and m: Plots show FSCA (blue) and FSVA (green) implementations with precomputed covariance matrices (dashed lines) and without (solid lines).

converge. It is a function of the spread of the eigenvalues of the covariance matrix $\mathbf{C}$, and hence problem dependent.

4 FSCA with Backward Refinement

Ideally we would like to find the subset of k columns of $\mathbf{X}$ (variables) that can optimally reconstruct $\mathbf{X}$, that is

$$\underset{\mathbf{Z}_k \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k)\mathbf{X}). \tag{25}$$

However, this is an NP hard combinatorial optimisation problem (requires the evaluation of $\binom{v}{k} = v!/((v-k)!k!)$ possible combinations of the variables). In general FSCA and other greedy local search approaches are sub optimal (i.e. they are not guaranteed to find the optimal subset of variables according to the defined optimization criteria), but they represent a pragmatic solution as searching over all possible subsets quickly becomes computationally intractable with increasing problem dimension.

As a consequence of the greedy strategy adopted by FSCA variables that are selected in early iterations of the algorithm can become redundant as other variables are included in later iterations. This can result in sub-optimal solutions and can also be detrimental to the performance of some applications, for example, the clustering application which will be presented in Section 6.3. To overcome this weakness, we propose introducing a backward refinement step similar to that presented in [25], [26] for forward selection regression applications, where, following completion of the forward selection process, selected variables are reviewed to see if they are still relevant and replaced if they are not.

Denoting $\mathbf{Z}_k^{(j)}(\mathbf{x}_i)$ as matrix $\mathbf{Z}_k$ with its j-th column replaced by $\mathbf{x}_i$, that is:

$$\mathbf{Z}_k^{(j)}(\mathbf{x}_i) = \mathbf{Z}_k + (\mathbf{x}_i - \mathbf{z}_j)\mathbf{e}_j^\mathsf{T}, \tag{26}$$

where $\mathbf{e}_j$ is a vector with its j-th element equal to 1 and all others elements equal to zero, we define $\mathbf{z}_j \in \mathbf{Z}_k$ as relevant if

$$V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k)\mathbf{X}) \geq \underset{\mathbf{x}_i \in \mathbf{X}/\mathbf{Z}_k}{max}\ V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k^{(j)}(\mathbf{x}_i))\mathbf{X}). \tag{27}$$

This backward refinement step can be performed either at step 5 or step 7 of the FSV implementation of the FSCA algorithm, as highlighted Algorithm 3. When placed at step 7 the refinement step is only undertaken once after the FSV algorithm has completed. In contrast, the refinement step is executed following the addition of each new variable if placed at step 5. We will refer to this latter implementation as recursive backward refinement.

There are also two flavours of the refinement step itself. In the first, referred to as Single-Pass Backward Refinement (SPBR) (summarised in Algorithm 4), the relevance of each variable is evaluated in turn moving sequentially through the variables from the oldest to the newest. In the second, to take account of the fact that variables that are initially relevant may become irrelevant following refinements to variables later in the sequence, the process is repeated until a complete pass occurs without any refinements taking place. This version of the algorithm (summarised in Algorithm 5) is referred to as Multi-Pass Backward Refinement (MPBR).

Note that by virtue of the sequencing of operations in each algorithm it follows that

$$V_{\mathbf{X}}(\hat{\mathbf{X}}_{FSCA}^k) \leq V_{\mathbf{X}}(\hat{\mathbf{X}}_{SPBR}^k) \leq V_{\mathbf{X}}(\hat{\mathbf{X}}_{MPBR}^k). \tag{28}$$

However, no such statement can be made with regard to R-SPBR or R-MPBR as they may follow different 'hill climbing' solution paths and hence it is possible for the solutions to be inferior to the non-recursive implementations when $k > 2$.

One of the side-effects of employing the backward refinement step is that it breaks the ordering of selected variables in terms of variance explained. If recovering this ordering is desirable, an additional modified FSV step can be performed on $\mathbf{Z}_k$ with respect to $\mathbf{X}$ after the refinement process has been completed (i.e. between Step 7 and 8 in Algorithm 3). As summarised in Algorithm 6, this involves recursively selecting the variables in $\mathbf{Z}_k$ based on how much of the variance of $\mathbf{X}$ that they explain.

Algorithm 4: $[\mathbf{Z}_k, r_c]$=SPBR($\mathbf{X}$,$\mathbf{Z}_k$)

Require: Forward selected variables $\mathbf{Z}_k$, data matrix $\mathbf{X}$
1: $r_c = 0$ (refinement count)
2: **for** $j = 1$ **to** $k-1$ **do**
3: $\quad i_j = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k^{(j)}(\mathbf{x}_i))\mathbf{X})$
4: $\quad$ **if** $\mathbf{z}_j \neq \mathbf{x}_{i_j}$ **then**
5: $\quad\quad r_c = r_c + 1$ (increment refinement count)
6: $\quad\quad \mathbf{Z}_k = \mathbf{Z}_k^{(j)}(\mathbf{x}_{i_j})$ (i.e. replace $\mathbf{z}_j$ with $\mathbf{x}_{i_j}$)
7: $\quad$ **end if**
8: **end for**
9: **if** $r_c > 0$ **then**
10: $\quad$ Repeat steps 3-7 for $j = k$
11: **end if**
12: **return** $\mathbf{Z}_k,\ r_c$

Fig. 5. Single-Pass Backward Refinement Algorithm

Algorithm 5: $[\mathbf{Z}_k]$=MPBR($\mathbf{X}$,$\mathbf{Z}_k$)

Require: Forward selected variables $\mathbf{Z}_k$, data matrix $\mathbf{X}$
1: $r_c = 1$ (refinement flag)
2: **while** $r_c > 0$ **do**
3: $\quad [\mathbf{Z}_k, r_c]$=SPBR($\mathbf{X}$,$\mathbf{Z}_k$)
4: **end while**
5: **return** $\mathbf{Z}_k$

Fig. 6. Multi-Pass Backward Refinement Algorithm

4.1 Computational complexity of backward refinement

The inclusion of backward refinement has major implications for the complexity of FSCA. The lowest complexity implementation is SPBR, which involves a combinatorial search of similar complexity to the basic FSV algorithm (eqt. 22), the only difference being that $\mathbf{Z}$ is now a fixed size matrix, that is $\mathbf{Z}_{(i)} \rightarrow \mathbf{Z}_k^{(j)}(\mathbf{x}_i)$, where $\mathbf{Z}_k^{(j)}(\mathbf{x}_i)$ is as defined in eqt. (26). Since the covariance matrix, and hence the $\mathbf{q}_{j(i)}$ terms, will have already been precomputed for the forward selection step, the only concern is the development of an efficient recursive update procedure for the inverse matrix $(\mathbf{Z}_k^{(j)}(\mathbf{x}_i)^\mathsf{T}\mathbf{Z}_k^{(j)}(\mathbf{x}_i))^{-1}$. This can be achieved by noting that

$$\mathbf{Z}_k^{(j)}(\mathbf{x}_i)^\mathsf{T}\mathbf{Z}_k^{(j)}(\mathbf{x}_i) = \mathbf{Z}_k^\mathsf{T}\mathbf{Z}_k + \mathbf{g}_{j(i)}\mathbf{e}_j^\mathsf{T} + \mathbf{e}_j\mathbf{h}_{j(i)}^\mathsf{T}, \quad (29)$$

where

$$\mathbf{g}_{j(i)} = \mathbf{Z}_k^\mathsf{T}(\mathbf{x}_i - \mathbf{z}_j), \quad (30)$$

$$\mathbf{h}_{j(i)} = \mathbf{g}_{j(i)} + (\mathbf{x}_i - \mathbf{z}_j)^\mathsf{T}(\mathbf{x}_i - \mathbf{z}_j)\mathbf{e}_j. \quad (31)$$

It then follows, by application of the matrix inversion lemma [44], specifically the Sherman-Morrison formula [45], that

$$(\mathbf{Z}_k^{(j)}(\mathbf{x}_i)^\mathsf{T}\mathbf{Z}_k^{(j)}(\mathbf{x}_i))^{-1} = \mathbf{A}_{j(i)} - \frac{\mathbf{A}_{j(i)}\mathbf{e}_j\mathbf{h}_{j(i)}^\mathsf{T}\mathbf{A}_{j(i)}}{1 + \mathbf{h}_{j(i)}^\mathsf{T}\mathbf{A}_{j(i)}\mathbf{e}_j}, \quad (32)$$

where

$$\mathbf{A}_{j(i)} = (\mathbf{Z}_k^\mathsf{T}\mathbf{Z}_k)^{-1} - \frac{(\mathbf{Z}_k^\mathsf{T}\mathbf{Z}_k)^{-1}\mathbf{g}_{j(i)}\mathbf{e}_j^\mathsf{T}(\mathbf{Z}_k^\mathsf{T}\mathbf{Z}_k)^{-1}}{1 + \mathbf{e}_j^\mathsf{T}(\mathbf{Z}_k^\mathsf{T}\mathbf{Z}_k)^{-1}\mathbf{g}_{j(i)}}. \quad (33)$$

This recursive inverse update can be computed in $\mathsf{O}(8k^2 + 4k + 6)$ flops and hence has $\mathsf{O}(8k^2)$ complexity, which compares favourably to the $\mathsf{O}(4k^2)$ complexity of the forward step inverse (eqt. 24). The overall additional complexity of executing SPBR is then $\mathsf{O}(2v^2k^3 + 8vk^3)$. In contrast, the recursive SPBR implementation contributes $\mathsf{O}(0.5v^2k^4 + 2vk^4)$ additional complexity.

Since repetition of the MPBR loop is dependent on refinements taking place in the previous pass, the number of repetitions and hence overall algorithm complexity of the multi-pass implementations cannot be determined *a priori*. If we denote the average number of repetitions as λ then their complexity can be expressed as λ times the complexity of the corresponding SPBR and recursive SPBR implementations. The optional reordering step has $\mathsf{O}(2vk^3)$ complexity. Hence, the overall algorithm complexity of FSVA with Backward refinement is

$$\mathsf{O}(v^2mk^2 + (2\lambda + \frac{2}{3})v^2k^3) \quad (34)$$

for non-recursive implementations and

$$\mathsf{O}(v^2mk^2 + \frac{\lambda}{2}v^2k^4 + \frac{2}{3}v^2k^3 + 2\lambda vk^4) \quad (35)$$

for recursive implementations, where $\lambda = 1$ corresponds to SPBR and $\lambda > 1$ MPBR.

Algorithm 6: $[\mathbf{Z}_k^o]$=ReOrder($\mathbf{X}$,$\mathbf{Z}_k$)

Require: Forward selected variables $\mathbf{Z}_k$, data matrix $\mathbf{X}$
1: $\mathbf{Z}_0^o = \emptyset$
2: **for** $j = 1$ **to** k **do**
3: $\quad i_j = \underset{\mathbf{z}_i \in \mathbf{Z}_k / \mathbf{Z}_{j-1}^o}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}([\mathbf{Z}_{j-1}^o\ \mathbf{z}_i])\mathbf{X})$
4: $\quad \mathbf{Z}_j^o = [\mathbf{Z}_{j-1}^o\ \mathbf{z}_{i_j}]$
5: **end for**
6: **return** $\mathbf{Z}_k^o$

Fig. 7. Modified FSV procedure for reordering variables following backward refinement

5 Simulated Datasets

In this section various simulated datasets are used to highlight the differences between FSCA and FSCA with backward refinement. The algorithms considered are:

- **FSCA**: Forward Selection Component Analysis (Algorithm 2 or 3)
- **SPBR**: Single-Pass Backward Refinement (Algorithm 3 with Algorithm 4 employed at step 7)
- **MPBR**: Multi-Pass Backward Refinement (Algorithm 3 with Algorithm 5 employed at step 7)
- **R-SPBR**: Recursive Single-Pass Backward Refinement (Algorithm 3 with Algorithm 4 employed at step 5)
- **R-MPBR**: Recursive Multi-Pass Backward Refinement (Algorithm 3 with Algorithm 5 employed at step 5)

Comparisons are also made with PCA, the Orthogonal Principal Feature Selection approximation to FSCA (OPFS), and Sparse PCA where appropriate.

5.1 Example 1: Four Distinct Variables

As a first example we define four base variables $\mathbf{w}_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0 \sim N(0,1)$, 20 noise variables $\epsilon_1, \ldots, \epsilon_{20} \sim N(0, 0.1)$ and two larger noise variables $\epsilon_{21}, \epsilon_{22} \sim N(0, 0.4)$. These variables are used to generate a subset of variables similar to $\mathbf{w}_0$: $\{\mathbf{w}_i = \mathbf{w}_0 + \epsilon_i\}_{i=1,\ldots,5}$, a subset of variables similar to $\mathbf{x}_0$: $\{\mathbf{x}_i = \mathbf{x}_0 + \epsilon_{i+5}\}_{i=1,\ldots,5}$, a subset of variables similar to $\mathbf{y}_0$: $\{\mathbf{y}_i = \mathbf{y}_0 + \epsilon_{i+10}\}_{i=1,\ldots,5}$, a subset of variables similar to $\mathbf{z}_0$: $\{\mathbf{z}_i = \mathbf{z}_0 + \epsilon_{i+15}\}_{i=1,\ldots,5}$ and two additional redundant variables defined as $\mathbf{h}_1 = \mathbf{w}_0 + \mathbf{x}_0 + \epsilon_{21}$ and $\mathbf{h}_2 = \mathbf{y}_0 + \mathbf{z}_0 + \epsilon_{22}$. The complete dataset is then defined as $\mathbf{X} = [\mathbf{w}_0, \ldots, \mathbf{w}_5, \mathbf{x}_0, \ldots, \mathbf{x}_5, \mathbf{y}_0, \ldots, \mathbf{y}_5, \mathbf{z}_0, \ldots, \mathbf{z}_5, \mathbf{h}_1, \mathbf{h}_2]$, with $\mathbf{X} \in \mathbb{R}^{m \times 26}$. Hence, by design the dataset is highly redundant, with only 4 of the 26 variables independent. As such, the information it contains can be optimally summarized by the 4 base variables $(\mathbf{w}_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0)$.

Table 2 shows the variance explained by PCA, OPFS, FSCA and the 4 backward refinement FSCA enhancements as a function of the number of selected variables k for an instance of the dataset with $m = 1000$, while Table 3 shows the sets of variables selected by FSCA, SPBR and OPFS for each value of k. Results for MPBR, R-SPBR and R-MPBR are omitted from Table 3 as they are identical to SPBR.

When FSCA is applied to this dataset in general the first FSV will be $\mathbf{h}_1$ and the second will be $\mathbf{h}_2$, or viva versa, as dictated by the noise realization. Subsequent selections are then from among the base variables until at $k = 6$ all 4 based variables are selected. Hence, the initial selections become redundant as additional variables are added. As expected the backward refinement algorithms explain greater variance than FSCA when $k = 3, 4$ and 5. Note, that at $k = 4$ the refinement of the FSCA solution by SPBR identifies the optimum set of variables (i.e. the 4 base variables), hence there is no scope for further improvement by the more advanced refinement algorithms.

For comparison purposes OPFS results are also presented in the tables. As can be seen, the variable selections for $k = 1$ and 2 are the same as FSCA, but thereafter OPFS takes a different path, and ends with a suboptimal solution at $k = 6$ (explained variance of 96% versus 99%). The performance of OPFS varies considerably for different instances of the dataset. This is illustrated in Figure 8, which shows the variation in performance of each method over 200 different dataset realizations with $m = 100$ when selecting $k = 4$ and 6 components. FSCA also shows considerable variation in performance but is in general superior to OPFS. A pairwise comparison of the variance explained by OPFS and FSCA over 1000 repetitions of the dataset shows that OPFS only outperforms FSCA 2% of the time. In contrast, SPBR and the other refinement algorithms are consistently superior to FSCA and OPFS and show little variation in performance over the different dataset realizations.

5.2 Example 2: Block Redundancy

In this example the dataset consists of a block of independent variables $\mathbf{X}^0$ augmented by a second block of noise

TABLE 2
Example 1: The percentage of explained variance achieved with PCA, OPFS, FSCA, and its backward refinement variants, for different numbers of selected components

k	PCA	OPFS	FSCA	SPBR	MPBR	R-SPBR	R-MPBR
1	30.41	25.88	25.88	25.88	25.88	25.88	25.88
2	56.68	54.34	54.34	54.34	54.34	54.34	54.34
3	80.47	75.72	75.82	78.11	78.11	78.11	78.11
4	98.60	93.62	93.80	98.22	98.22	98.22	98.22
5	99.03	96.36	96.56	98.78	98.78	98.78	98.78
6	99.43	96.40	99.31	99.31	99.31	99.31	99.31

TABLE 3
Example 1: Variables selected at each step by FSCA, SPBR and OPFS

k	FSCA	SPBR	OPFS
1	$\{\mathbf{h}_1\}$	$\{\mathbf{h}_1\}$	$\{\mathbf{h}_1\}$
2	$\{\mathbf{h}_1, \mathbf{h}_2\}$	$\{\mathbf{h}_1, \mathbf{h}_2\}$	$\{\mathbf{h}_1, \mathbf{h}_2\}$
3	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{x}_0\}$	$\{\mathbf{w}_0, \mathbf{h}_2, \mathbf{x}_0\}$	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{w}_0\}$
4	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{x}_0, \mathbf{z}_0\}$	$\{\mathbf{w}_0, \mathbf{y}_0, \mathbf{x}_0, \mathbf{z}_0\}$	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{w}_0, \mathbf{z}_0\}$
5	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{x}_0, \mathbf{z}_0, \mathbf{w}_0\}$	$\{\mathbf{y}_0, \mathbf{h}_1, \mathbf{x}_0, \mathbf{z}_0, \mathbf{w}_0\}$	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{w}_0, \mathbf{z}_0, \mathbf{y}_0\}$
6	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{x}_0, \mathbf{z}_0, \mathbf{w}_0, \mathbf{y}_0\}$	$\{\mathbf{x}_0, \mathbf{y}_0, \mathbf{w}_0, \mathbf{z}_0, \mathbf{h}_1, \mathbf{h}_2\}$	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{w}_0, \mathbf{z}_0, \mathbf{y}_0, \mathbf{w}_2\}$

perturbed redundant variables $\mathbf{X}^1$ generated as a linear combination of the variables in $\mathbf{X}^0$. In particular we define:

- $\mathbf{X}^0 \in \mathbb{R}^{n \times u}$: $\mathbf{X}^0_{i,j} \sim N(0,1)$
- $\phi \in \mathbb{R}^{u \times (v-u)}$: $\phi_{i,j} \sim N(0,1)$
- $\epsilon \in \mathbb{R}^{n \times (v-u)}$: $\epsilon_{i,j} \sim N(0, 0.1)$
- $\mathbf{X}^1 = \mathbf{X}^0 \cdot \phi + \epsilon$
- $\mathbf{X} = [\mathbf{X}^0, \mathbf{X}^1]$

We generated 1000 instances of this dataset for different values of u and v for $n = 200$ and in each case computed a $k = u$ components FSCA, SPBR, R-SPBR, MPBR and R-MPBR. The variance explained by the k components selected by each algorithm averaged over the 1000 repetitions is reported in Table 4. The table also reports S_c, the percentage of true variables selected by each method, defined as

$$S_c = |\{\mathbf{z}_1, \ldots \mathbf{z}_u\} \cap \{\mathbf{x}_1, \ldots, \mathbf{x}_u\}| / u \times 100. \quad (36)$$

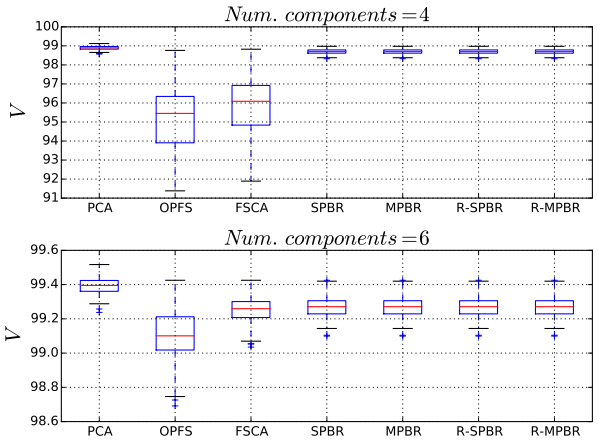

Fig. 8. Example 1: Boxplots showing variation in performance of each method for $k = 4$ and 6 components, over 200 Monte Carlo repetitions

TABLE 4
Example 2: Percentage variance explained ($V_{\mathbf{X}}$) and percentage of true variables selected (S_c) with FSCA and its backward refinement variants (averaged over 1000 repetitions)

Percentage of variance explained ($V_{\mathbf{X}}$)						
u	v	FSCA	SPBR	MPBR	R-SPBR	R-MPBR
10	30	99.75	99.87	99.89	99.88	99.89
15	50	99.77	99.89	99.92	99.90	99.92
20	75	99.78	99.90	99.94	99.90	99.94
25	100	99.78	99.91	99.94	99.91	99.94
Percentage of true variables selected (S_c)						
u	v	FSCA	SPBR	MPBR	R-SPBR	R-MPBR
10	30	22.38	48.80	70.11	47.73	66.30
15	50	16.03	43.73	74.20	41.98	72.06
20	75	14.70	41.60	81.66	45.11	79.82
25	100	12.85	34.74	71.46	38.21	71.95

As expected, the introduction of a refinement step consistently increases the explained variance relative to FSCA with SPBR reducing unexplained variance by 50%-64% and MPBR reducing it by 58%-73%. There is no appreciable difference between the performance of the recursive and non-recursive implementations of each algorithm. A similar pattern is observed with respect to the number of true variables selected by each method, FSCA: 12%-22%, SPBR/R-SPBR: 35%-49% and MPBR/R-MPBR: 66%-82%.

Noting that PCA provides an upper bound on achievable explained variance for a given number of components, Figure 9 shows the variance explained by FSCA and the various backward refinement algorithms with $k = 1, 2, ..., 12$ selected components for the case where $u = 10$, $v = 30$ and $n = 1000$, expressed as a percentage of the variance explained by the equivalent number of PCs obtained using PCA. As can be seen, SPBR consistently provides improved performance over FSCA for $k > 1$. For values of k in the vicinity of the true dimensionality of the data, MPBR is marginally superior to SPBR (57.5% versus 49.2% reduction in unexplained variance at $k = 10$, 11.8% versus 9.7% at $k = 8$ and 27.0% versus 23.9% at $k = 12$). In general the improvement due to backward refinement decreases rapidly as k increases beyond the true dimensionality of the data.

5.3 Example 3: Sparse PCA Dataset

This example is a simulated dataset used in [19] to assess the performance of the sparse PCA algorithm introduced therein. The dataset is generated by 3 hidden variables $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$

- $\mathbf{v}_1 \sim N(0, 290)$ $\mathbf{v}_2 \sim N(0, 300)$.
- $\mathbf{v}_3 = -0.3\mathbf{v}_1 + 0.952\mathbf{v}_2 + \epsilon$ where $\epsilon \sim N(0, 1)$.

and 10 observed variables

- $\mathbf{x}_i = \mathbf{v}_1 + \epsilon_i^1$ where $\epsilon_i^1 \sim N(0, 1)$ $for\ i = 1, \ldots, 4$
- $\mathbf{x}_i = \mathbf{v}_2 + \epsilon_i^2$ where $\epsilon_i^2 \sim N(0, 1)$ $for\ i = 5, \ldots, 8$
- $\mathbf{x}_i = \mathbf{v}_3 + \epsilon_i^3$ where $\epsilon_i^3 \sim N(0, 1)$ $for\ i = 9, 10$

The final data matrix $\mathbf{X} \in \mathbb{R}^{n \times 10}$ is then defined as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{10}]$, where $n = 1000$ is the number of samples.

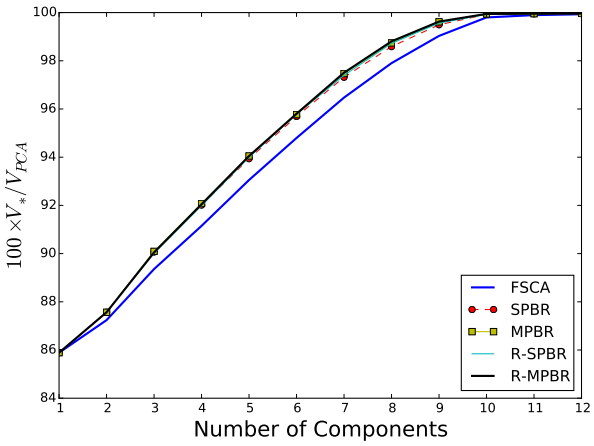

Fig. 9. Example 2: The percentage of variance explained as a function of the number of selected components for $u = 10, v = 30$

TABLE 5
Example 3: The 1st and 2nd loading generated by PCA and SPCA (λ = 20) and the 1st and 2nd FSC obtained with FSCA and SPBR

i	PC_1	PC_2	SPC_1	SPC_2	FSC_1	FSC_2	$SPBR_1$	$SPBR_2$
1	-0.13	0.48		-80.00				
2	-0.13	0.48		-80.00				
3	-0.13	0.48		-80.00		1		1
4	-0.13	0.48		-80.00				
5	0.39	0.16	79.61				1	1
6	0.39	0.16	79.61					
7	0.39	0.16	79.61					
8	0.39	0.16	79.61					
9	0.41	0.01	77.43	3.09	1	1		
10	0.41	0.01	77.43	3.09				
$V_{\mathbf{X}}$	60.80	99.99	59.43	99.99	60.75	99.99	58.22	99.99

The results reported in Table 5, which are for a single realization of the dataset, show that both FSCA and SPBR with 2 components explain more than 99% of the total variance. In general, for other realizations FSCA will always select either $\mathbf{x}_9$ or $\mathbf{x}_{10}$ as one of the two variables. SPBR and MPBR will instead select one variable from the group generated by $\mathbf{v}_1$ and one of from the group generated by $\mathbf{v}_2$. PCA and SPCA assign similar values to similar variables due to the grouping effect. However, as a result they do not omit redundant variables. Thus, while SPCA yields sparse solutions it is not the optimal choice if the objective is to select a compact set of variables to represent the data.

6 Application Examples

6.1 Pitprops Dataset

The pitprops dataset, originally introduced by [46] as a PCA case study, is a widely used benchmark problem for evaluating the performance of PCA and SPCA like methods. The dataset consists of 180 samples of 13 variables describing properties of timber and was used by the British Forestry Commission in a study to establish if home-grown timber had sufficient strength to be used to provide roof support struts 'Pitprops' for mines. Using the correlation matrix for the dataset provided in [46] we generated 180 samples of a multivariate normal distribution to synthesise an approximation of the original dataset.

TABLE 6
Pitprops dataset: Percentage of explained variance as a function of the number of variables selected for each algorithm

k	PCA	SPCA	FSCA	SPBR	MPBR	R-SPBR	R-MPBR
3	67.41	36.16	55.98	60.04	60.04	60.04	60.04
4	75.52	60.86	67.13	67.13	67.13	68.33	68.33
5	82.13	65.40	74.07	75.76	75.76	75.76	75.76
6	88.00	71.65	80.18	82.38	82.38	82.38	82.38
7	92.33	72.59	85.73	86.67	87.89	87.89	87.89
7	92.33	77.43	85.73	86.67	87.89	87.89	87.89
9	97.63	85.31	94.52	95.87	95.87	95.87	95.87
11	99.38	93.05	98.79	98.79	98.79	98.79	98.79
11	99.38	95.11	98.79	98.79	98.79	98.79	98.79
12	99.72	96.89	99.41	99.41	99.41	99.41	99.41
13	100.0	100.0	100.0	100.0	100.0	100.0	100.0

TABLE 7
Pitprops dataset: The 6 variables selected by each algorithm and the optimum set of 6 variables (BEST) in terms of maximizing the percentage of explained variance in the dataset

FSCA	SPBR	MPBR	R-SPBR	R-MPBR	BEST
whorls	ringbut	ringbut	ringbut	ringbut	ringbut
moist	moist	moist	moist	moist	moist
length	length	length	length	length	length
ringtop	clear	clear	clear	clear	clear
clear	bowmax	bowmax	bowmax	bowmax	bowmax
ovensg	ovensg	ovensg	ovensg	ovensg	ovensg
V_X					
80.18	82.38	82.38	82.38	82.38	82.38

For the synthesised dataset six SPCA components were computed for a range of different values of the penalty λ. For each value of λ the number of uniquely selected variables was identified and then the corresponding number of FSVs computed with FSCA, SPBR, R-SPBR, MPBR and R-MPBR. The number of variables selected and the percentage of explained variance are reported in Table 6. In order to provide an upper bound for the percentage of explained variance that can be achieved we have also reported the corresponding values obtained with PCA. From the results it can be observed that SPCA is the method that explains the least variance for a given number of selected variables.

In SPCA the number of selected variables is indirectly chosen through a penalty parameter λ. In some situations it can be difficult to choose λ to select a specific number of variables. In our case, for example, it has not been possible to select $1, 2, 8$ or 13 variables using SPCA. In particular, observe that for 7 and 11 selected variables SPCA returns 2 different results. This is due to the fact that two different subsets of $7/11$ variables were returned for two different values of λ. Table 7 lists the variables selected by a 6 components FSCA, SPBR, MPBR, R-SPBR and R-MPBR together with the set of 6 variables that maximize the percentage of explained variance (BEST). This set has been determined by evaluating all possible subsets of 6 variables. Observe that MPBR, R-SPBR and R-MPBR select the same variables as BEST while SPBR replaces the variable 'moist' with 'testsg'. In contrast, FSCA selects only 3 variables in common with BEST.

6.2 Plasma Etch OES Analysis

In semiconductor manufacturing Optimal Emission Spectroscopy (OES) is increasingly used to monitor plasma etch processes. Due to the high dimensionality and correlated nature of OES data dimensionality reduction techniques such as PCA are usually employed as a pre-processing step (see for example [12], [23], [47] and [48]). Here we employ a sample OES dataset collected during the processing of a single wafer as a case study for comparing the orthogonal decompositions generated by FSCA and PCA. The OES spectrum in question, plotted in Fig. 10, consists of optical emission intensity time series data for each of the 2000 active spectrometer channels (each channel corresponds to a different optical wavelength in the range 192-875 nm). Each time series has 55 samples, hence the resulting dataset is a matrix $\mathbf{X} \in \mathbb{R}^{55\times 2000}$ of intensity values.

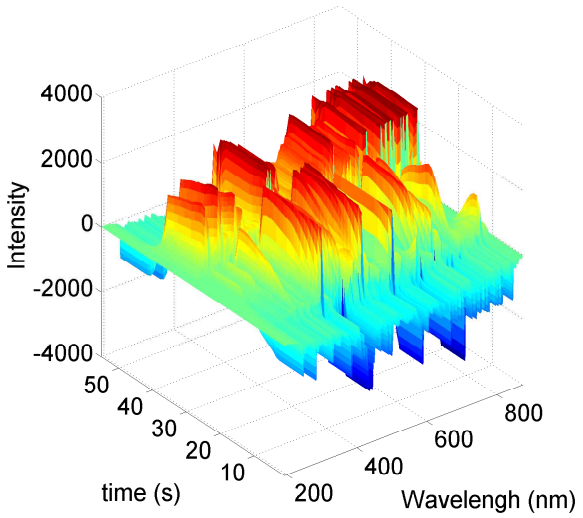

Fig. 10. Plasma Etch Process OES Spectrum

TABLE 8
Plasma Etch: Accumulative variance explained by PCA, FSCA and the four backward refinement variants of FSCA for different values of k

k	PCA	FSVA	SPBR	MPBR	R-SPBR	R-MPBR
1	77.04	76.63	76.63	76.63	76.63	76.63
2	96.53	96.00	96.37	96.37	96.37	96.37
3	98.20	98.00	98.14	98.14	98.14	98.14
4	99.47	99.27	99.42	99.42	99.42	99.42
5	99.69	99.50	99.65	99.65	99.65	99.65
6	99.81	99.66	99.75	99.79	99.79	99.79
7	99.88	99.79	99.85	99.86	99.87	99.86
8	99.93	99.89	99.91	99.92	99.92	99.92
9	99.95	99.93	99.94	99.95	99.95	99.95

The highly correlated nature of the data is evident from Table 8, which shows that the first 4 PCs and the first 4 variables selected by FSCA and its backward refinement variants explain more than 99% of the variation in data. In Fig. 11 the PCA loadings and scores are compared with the FSVs and FSCs obtained with FSCA. This reveals that the etch process can be analysed using either PCA scores or FSCA components. In particular, observe that the FSCA

components and PCA scores tend to have similar trends. As noted previously, the PCA scores are obtained as a linear combination of all 2000 original variables (as defined by the PCA loadings), while the four FSCs can be expressed as a linear combination of just 4 original variables (the FSVs). The benefit of being able to trace process variably back to a small number of OES wavelengths is that individual wavelengths map to specific chemical species present in the plasma, enabling process engineers to gain insight into the underlying drivers of process variability.

The computation time in seconds for each algorithm in reported in Table 9 for different numbers of computed components/variables selected. As expected, computational time grows rapidly with increasing k for the more complex refinement algorithms. For example, while SPBR is only twice as computationally intensive as FSVA at $k = 9$ R-MPBR is more than 20 times more computationally intensive.

TABLE 9
Plasma Etch: Computation time (in seconds) for PCA, FSVA and the four backward refinement variants of FSCA for different values of k

k	PCA	FSVA	SPBR	MPBR	R-SPBR	R-MPBR
1	0.03	0.05	0.10	0.10	0.10	0.10
2	0.04	0.10	0.20	0.31	0.25	0.36
3	0.05	0.16	0.32	0.50	0.47	0.75
4	0.07	0.22	0.46	0.96	0.80	1.32
5	0.09	0.28	0.60	1.25	1.16	2.32
6	0.10	0.36	0.76	2.40	1.67	3.25
7	0.13	0.42	0.93	3.54	2.22	5.85
8	0.14	0.49	1.09	2.91	2.90	8.35
9	0.15	0.56	1.29	3.47	3.63	11.86

6.3 Wafer Site Optimisation

As a final application example, we evaluate the performance of SPBR, MPBR, R-SPBR and R-MPBR as alternatives to FSCA for the semiconductor wafer metrology site optimisation methodology developed in [24]. The pertinent details are as follows. The objective of the methodology is to use historical metrology data for a set of candidate measurement sites to determine the minimum set of sites that need to be measured in order to accurately reconstruct wafer profiles. The case study dataset consists of production metrology data for a deposition process used in the manufacture of read-write heads, a key component of hard disk drives. The dataset, which was collected over several weeks from a single production tool for the process, contains measurements of 50 candidate sites for 316 wafers. Hence, $\mathbf{X} \in \mathbb{R}^{316\times 50}$ and the site selection problem equates to selecting the subset of columns of $\mathbf{X}$ that best describe $\mathbf{X}$. For a detailed description of the problem statement, solution methodology and case study dataset the reader is referred to [24].

Here, FSCA and the newly proposed backward refinement variants are employed to determine the optimum subset of wafer sites. The percentage of variance explained by each method for different numbers of selected sites (k) is reported in Table 10. Defining 99% variance explained as the minimum reconstruction accuracy threshold, it follows that 7 sites are needed when using FSC, while 6 sites are sufficient when using SPBR, MPBR, R-SPBR and R-MPBR.

TABLE 10
Wafer sites: The percentage of variance explained by the various methods for different values of k, the number of selected wafer sites

k	PCA	FSCA	SPBR	MPBR	R-SPBR	R-MPBR
1	42.69	38.84	38.84	38.84	38.84	38.84
2	68.69	64.44	67.01	67.01	67.01	67.01
3	85.72	82.59	84.57	85.08	84.79	85.08
4	98.48	96.37	97.40	97.58	97.58	97.58
5	**99.12**	97.59	98.63	98.74	98.72	98.73
6	99.47	98.76	**99.19**	**99.20**	**99.17**	**99.16**
7	99.67	**99.22**	99.45	99.46	99.42	99.39
8	99.75	99.47	99.60	99.60	99.59	99.58
9	99.81	99.64	99.71	99.71	99.71	99.69
10	99.86	99.72	99.77	99.80	99.78	99.79

The PCA results, which are also recorded in Table 10, show that the lower bound on the number of sites required is 5.

A plot of the variance explained by each of the FSCA based methods as a percentage of the variance explained by PCA is given in Fig. 12. Again in this example we observe that, as expected, SPBR outperforms FSCA and MPRB outperforms SBBR (to a lesser extent), but that unlike the previous examples, R-SPBR and R-MPBR are sometimes marginally inferior to their non-recursive counterparts (i.e. for $k \geq 5$).

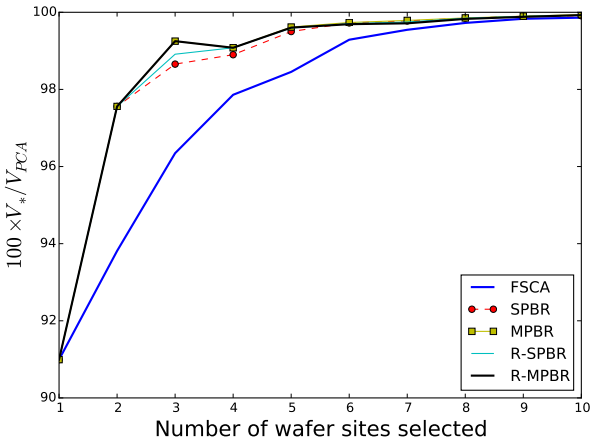

Fig. 12. Wafer sites: The variance explained by the metrology sites selected by FSCA and the 4 backward refinement algorithms for different values of k expressed as a percentage of the variance explained by the equivalent number of PCs

It is also interesting to observe how representative the FSCA selected sites are of the full wafer surface. This can be visualised by clustering the unmeasured sites in k clusters $\mathcal{C}_{\mathbf{z}_1}, \ldots, \mathcal{C}_{\mathbf{z}_k}$ according to their similarity to the k FSCA selected sites. Here, the similarly between an FSCA site, $\mathbf{z}_i$, and an unmeasured site, $\mathbf{x}_j$, is defined in terms of the impact on reconstruction accuracy of replacing $\mathbf{z}_i$ with $\mathbf{x}_j$. Specifically, denoting the k selected variables as $\mathbf{Z}_k = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$, and noting the definition of $\mathbf{Z}_k^{(i)}(\mathbf{x}_j)$ given in eqt. (26), sites are assigned to clusters according to the rule

$$\mathbf{x}_j \in \mathcal{C}_{\mathbf{z}_i} \text{ if } i = \underset{p=1\ldots k}{argmax}\, V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k^{(p)}(\mathbf{x}_j))\mathbf{X}). \quad (37)$$

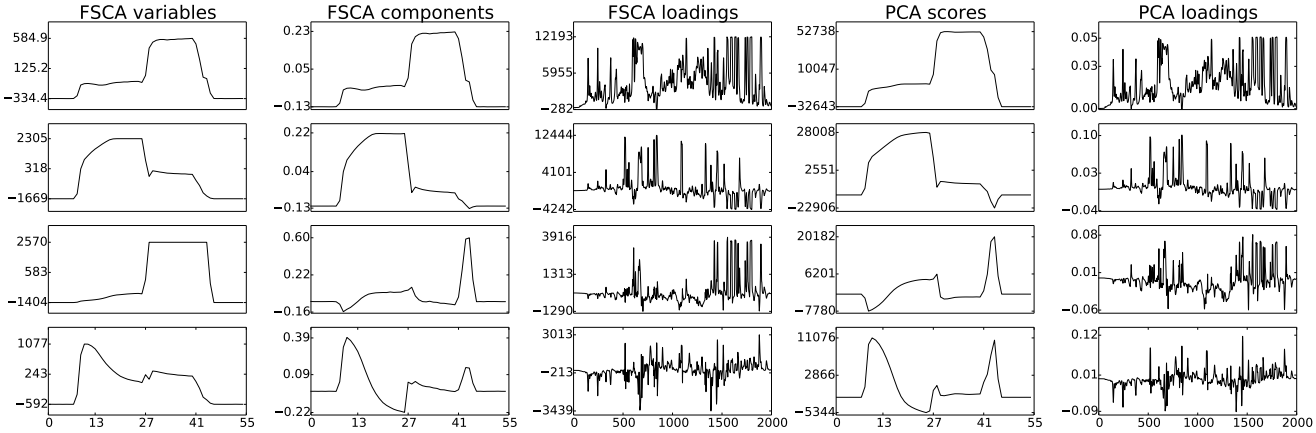

Fig. 11. Plasma Etch Process OES dataset: The first 4 PCA and FSCA components and scores

Fig. 13 shows the clusters obtained with each of the FSCA algorithms for $k = 4$ and $k = 8$. The clusters are represented by markers of different colour and/or shape. Of particular note is the variation in spatial consistency of clusters. It is apparent that the refinement steps yield much better spatial consistency of clusters than FSCA, with the biggest improvements occurring with SPBR. Using FSCA 50% of the clusters are fragmented for both $k = 4$ and $k = 8$, while using SPBR only 1 cluster (25%) is fragmented when $k = 4$ and none are fragmented when $k = 8$. MPBR, R-SPBR and R-MPBR yield similar results to SPBR with some minor variation at the boundaries between clusters. It is also noteworthy that in the FSCA plot for $k = 8$ the 'black star' site is clustered with only one other site which is in a spatially unrelated area of the wafer. These anomalies are a consequence of the sites initially selected by FSCA becoming redundant as additional sites are selected, as discussed in Section 4. This issue, which detracts from the interpretability of clusters, is addressed through the introduction of the backward refinement step.

7 Discussion and Conclusions

This paper has sought to provide a comprehensive presentation of Forward Selection Component Analysis, as the unsupervised counterpart of Forward Selection Regression and an alternative to PCA for dimensionality reduction and variable selection in large highly correlated datasets. A number of alternative FSCA algorithm implementations have been described, namely FSCA and FSVA, with and without pre-computation of the covariance matrix, and their computational complexity analysed. In particular, this analysis reveals that: (1) all algorithms scale linearly with the number of measurements m and quadratically with the number of variables v; (2) FSCA implementations grow linearly with the number of selected variables k, while FSVA implementations grow cubically with k; and (3) the optimum choice of implementation is dependent on the ratio $k/\sqrt{m}$. In general, it is computationally advantageous to pre-compute and store the covariance matrix when using either FSCA or FSVA, with FSVA computationally the most efficient implementation provided $k/\sqrt{m} < \sqrt{1.5}$. FSCA without pre-computation of the covariance matrix is the superior implementation when $k/\sqrt{m} > \sqrt{3}$.

A number of novel backward refinement variants of FSCA have also been proposed and efficient algorithm implementations developed. Results from simulated and application case studies confirm that the refinements yield improvements in performance relative to FSCA in terms of variance explained for a given number of components/variables selected, better variable selection and, in the case of the wafer site optimisation problem, more coherent FSCA clusters. Overall the key observations are that MPBR is superior to SPBR, which is in turn superior to FSCA, and that there is little, if any, benefit to be gained from employing the recursive formulations (R-SPBR and R-MPBR) over their non-recursive counterparts. Indeed, in some instances the recursive implementations can yield poorer results.

In terms of computational complexity the ordering is FSCA $<$ SPBR $<$ MPBR $<$ R-SPBR $<$ R-MPBR, with SPBR having the same asymptotic complexity as FSCA. It is also noteworthy that the largest relative improvement in performance in terms of variance explained occurs with the change from FSCA to SPBR. As such, for most practical applications SPBR is recommended as it provides a good balance between complexity and quality of results.

In all the case studies presented in the paper algorithm performance is considered in an unsupervised context, as this is the natural framework for comparing unsupervised variable selection techniques. The interested reader is referred to Appendix B for a supplementary linear regression case study where performance is also assessed in a supervised context. Specifically, FSCA and its variants are employed to select the model inputs from a large candidate set, and performance is evaluated in terms of the prediction capability of the resulting models.

Acknowledgments

The authors would like to thank Adrian Johnston, Seagate Technology and Niall MacGearailt, Intel Ireland for the provision of use cases and Maynooth University for the financial support provided. The authors would also like to acknowledge the anonymous reviewers for their valuable suggestions which have greatly enhanced the paper.

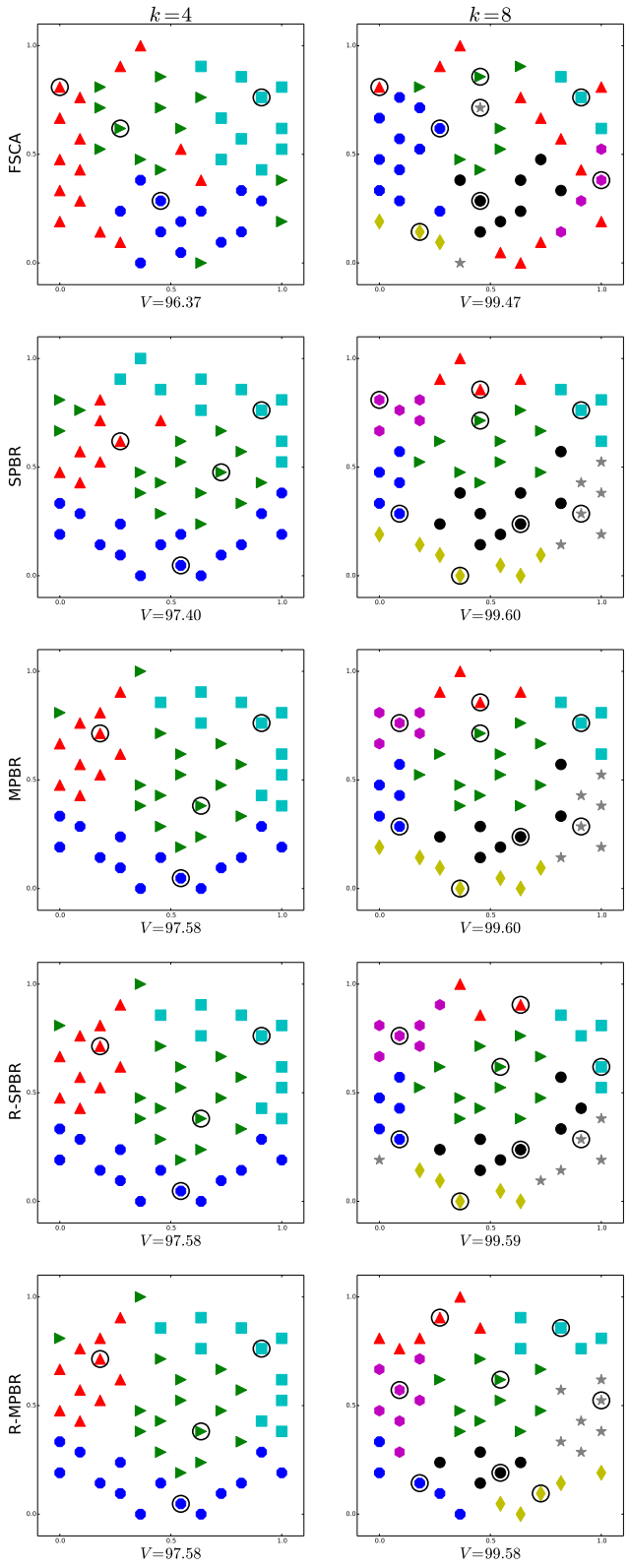

Fig. 13. The FSCA clusters obtained with FSCA, SPBR, MPBR, R-SPBR and R-MPBR for $k = 4$ and $k = 8$. In each case the FSCA selected sites are indicated by circles and the associated clusters by markers of different colour and/or shape. The percentage variance explained by the different algorithms is reported under each plot.

APPENDIX A: PROPERTIES OF $V_{\mathbf{X}}(.)$

Theorem 1. Given projection matrix $\Phi(\mathbf{S})$ as defined in eqt. (4) and $V_{\mathbf{X}}(.)$ as defined in eqt. (6), if $\hat{\mathbf{X}} = \Phi(\mathbf{S})\mathbf{X}$ then $V_{\mathbf{X}}(\hat{\mathbf{X}}) \geq 0 \; \forall \mathbf{S} \in \mathbb{R}^{n \times k}$.

Proof: Choosing $\hat{\mathbf{X}} = \Phi(\mathbf{S})\mathbf{X}$ is equivalent to choosing $\hat{\mathbf{X}} = \mathbf{S\Theta}$, where $\mathbf{\Theta}$ is the solution to the convex minimization problem in eqt. (2). By definition, the $\mathbf{\Theta}$ that minimizes eqt. (2) maximizes $V_{\mathbf{X}}(\hat{\mathbf{X}})$, that is:

$$\mathbf{\Theta} = \underset{\tilde{\mathbf{\Theta}}}{argmax} V_{\mathbf{X}}(\mathbf{S}\tilde{\mathbf{\Theta}})$$

It then follows that $V_{\mathbf{X}}(\mathbf{S\Theta}) \geq V_{\mathbf{X}}(\mathbf{S}\tilde{\mathbf{\Theta}}) \; \forall \tilde{\mathbf{\Theta}}$. Choosing $\tilde{\mathbf{\Theta}}$ as the zero matrix $\mathbf{0}$ gives $\hat{\mathbf{X}} = \mathbf{0}$ and hence $V_{\mathbf{X}}(\mathbf{0}) = 0$. Therefore $V_{\mathbf{X}}(\mathbf{S\Theta}) \geq V_{\mathbf{X}}(\mathbf{0}) = 0$.

□

Theorem 2. If $V_{\mathbf{X}}(.)$ is defined as in eqt. (6) and $\Phi(.)$ defined as in eqt. (4) then

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \; V_{\mathbf{X}}(\Phi(\mathbf{x}_i)\mathbf{X}) = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \frac{\mathbf{x}_i \mathbf{X}\mathbf{X}^T \mathbf{x}_i}{\mathbf{x}_i^T \mathbf{x}_i}.$$

Proof: It immediately follows from eqt.(6) that

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \; V_{\mathbf{X}}(\hat{\mathbf{X}}) = \underset{\mathbf{x}_i \in \mathbf{X}}{argmin} \parallel \mathbf{X} - \hat{\mathbf{X}} \parallel_F^2,$$

where $\hat{\mathbf{X}} = \Phi(\mathbf{x}_i)\mathbf{X}$. Expressing the F-norm in terms of the trace operator gives

$$\begin{aligned} \parallel \mathbf{X} - \hat{\mathbf{X}} \parallel_F^2 &= tr((\mathbf{X} - \hat{\mathbf{X}})(\mathbf{X} - \hat{\mathbf{X}})^T) \\ &= tr(\mathbf{X}\mathbf{X}^T) + tr(\hat{\mathbf{X}}\hat{\mathbf{X}}^T) - 2tr(\mathbf{X}\hat{\mathbf{X}}) \\ &= tr(\mathbf{X}\mathbf{X}^T) + tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T\Phi(\mathbf{x}_i)) - 2tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T), \end{aligned}$$

where the last equivalence is obtained by replacing $\hat{\mathbf{X}}$ with $\Phi(\mathbf{x}_i)\mathbf{X}$ and noting that $\Phi(\mathbf{x}_i) = \Phi(\mathbf{x}_i)^T$. By application of the cyclic property of the trace operator and observing that $\Phi(\mathbf{x}_i)^2 = \Phi(\mathbf{x}_i)$ we can write $tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T\Phi(\mathbf{x}_i)) = tr(\Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T) = tr(\mathbf{X}\mathbf{X}^T)$, and therefore

$$\parallel \mathbf{X} - \hat{\mathbf{X}} \parallel_F^2 = 2tr(\mathbf{X}\mathbf{X}^T) - 2tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T).$$

It immediately follows that

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmin} \parallel \mathbf{X} - \hat{\mathbf{X}} \parallel_F = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \; tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T).$$

Finally, by application of the definition of $\Phi(.)$ in eqt. (4) and the properties of trace, the r.h.s. can be rewritten as

$$\begin{aligned} \underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \; tr(\frac{\mathbf{x}_i \mathbf{x}_i^T}{\mathbf{x}_i^T \mathbf{x}_i}\mathbf{X}\mathbf{X}^T) &= \underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \; tr(\frac{\mathbf{x}_i^T}{\mathbf{x}_i \mathbf{x}_i^T}\mathbf{X}\mathbf{X}^T\mathbf{x}_i) \\ &= \underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \frac{\mathbf{x}_i^T \mathbf{X}\mathbf{X}^T \mathbf{x}_i}{\mathbf{x}_i^T \mathbf{x}_i}. \end{aligned}$$

□

TABLE 11
Mean and (standard deviation) of the CV-NMSE (%) achieved with the regression models generated using PCA, FSCA, SPBR, MPBR, R-SPBR and R-MPBR for input selection

k	5	10	15
PCA	1.40 (0.05)	1.26 (0.04)	1.13 (0.04)
FSCA	1.45 (0.08)	1.25 (0.04)	**1.10** (0.05)
SPBR	1.36 (0.05)	1.26 (0.06)	1.11 (0.03)
MPBR	**1.35** (0.05)	1.22 (0.09)	1.12 (0.04)
R-SPBR	1.36 (0.05)	1.20 (0.07)	1.13 (0.04)
R-MPBR	1.36 (0.05)	**1.19** (0.06)	1.13 (0.04)

TABLE 12
Percentage of variance in $\mathbf{X}$ explained by the variables/features selected by PCA, FSCA, and its backward refinement variants

k	5	10	15
PCA	97.44	99.31	99.81
FSCA	96.16	98.80	99.57
SPBR	96.74	99.04	99.68
MPBR	96.74	99.07	99.70
R-SPBR	96.74	99.06	99.70
R-MPBR	96.74	99.07	99.70

APPENDIX B: A REGRESSION EXAMPLE

As a supplementary example we consider the application of the various FSCA techniques to input selection in a regression problem and benchmark their performance against PCA. Since, in general, unsupervised variable or feature selection techniques are not guaranteed to identify good predictors of a target output, and it is straightforward to design pathological examples where each method will perform poorly, to provide a fair comparison we select a dataset for a practical application where PCA yields good results. The application in question is the prediction of etch rate of a plasma etch process from optical emission spectroscopy (OES) measurements recorded from the etching chamber during wafer processing. The associated dataset, described fully in [49], is defined by an input matrix $\mathbf{X} \in \mathbb{R}^{2194\times 200}$ of OES summary statistics and an output vector $\mathbf{y} \in \mathbb{R}^{2194}$ of the recorded etch rate for 2194 production wafers.

A Monte Carlo study was undertaken in which the data was randomly split into training and test data, corresponding respectively to 70% and 30% of the data. Using the training data, k inputs were generated in turn with PCA, FSCA, SPBR, MPBR, R-SPBR, and R-MPBR, and in each case a linear regression model estimated. The performance of the models was evaluated by cross-validation on the test data. This process was repeated 20 times for $k = 5, 10$ and 15, and the mean and standard deviation of the normalized mean squared prediction error (denoted as the CV-NMSE) computed in each case. These values are reported in Table 11.

For completeness, the percentage of the variance in the input data explained by the selected input features/variables (as computed on the training data set) are reported in Table 12. The pattern is the same as observed in previous examples. PCA yields the highest variance explained and FSCA the lowest among the methods considered for each value of k. The backward refinement methods fall between these two extremes in terms of their performance, with SPBR yielding most improvement relative to FSCA.

Reviewing Table 11 it is evident that the inputs selected by all methods yield good prediction models with the CV-NSME ranging from 1.35% to 1.45% when $k = 5$, 1.19% to 1.26% when $k = 10$, and 1.10% to 1.13% when $k = 15$. It is interesting to note that while PCA explains the most variance in terms of the input data, MPBR yields the best regression model when $k = 5$, R-MPBR when $k = 10$ and FSCA when $k = 15$. While the differences in CV-NMSE are statistically significant at a 95% confidence level, there is no clear winner among the methods. This can be attributed to the absence of causality between unsupervised input variable/feature selection and selecting inputs that yield the best regression results. Overall, we can conclude that FSCA and its backward refinement variants are as effective as PCA for this application, with the added advantage of providing an easy to interpret model.

REFERENCES

[1] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[2] A. Miller, *Subset selection in regression*. CRC Press, 2012.

[3] R. Díaz-Uriarte and S. A. De Andres, "Gene selection and classification of microarray data using random forest," *BMC Bioinformatics*, vol. 7, no. 1, p. 3, 2006.

[4] X. Geng, D.-C. Zhan, and Z.-H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 6, pp. 1098–1107, 2005.

[5] P. Mitra, C. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 301–312, 2002.

[6] J. G. Dy and C. E. Brodley, "Feature selection for unsupervised learning," *The Journal of Machine Learning Research*, vol. 5, pp. 845–889, 2004.

[7] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proceedings of the 24th international conference on Machine Learning*. ACM, 2007, pp. 1151–1157.

[8] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[9] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[10] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[11] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[12] B. Flynn and S. McLoone, "Max separation clustering for feature extraction from optical emission spectroscopy data," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 24, no. 4, pp. 480–488, 2011.

[13] L. Mariey, J. Signolle, C. Amiel, and J. Travert, "Discrimination, classification, identification of microorganisms using ftir spectroscopy and chemometrics," *Vibrational Spectroscopy*, vol. 26, no. 2, pp. 151–159, 2001.

[14] J. Trygg, E. Holmes, and T. Lundstedt, "Chemometrics in metabonomics," *Journal of Proteome Research*, vol. 6, no. 2, pp. 469–479, 2007.

[15] G. P. McCabe, "Principal variables," *Technometrics*, vol. 26, no. 2, pp. 137–144, 1984.

[16] J. Cadima and I. T. Jolliffe, "Loading and correlations in the interpretation of principle compenents," *Journal of Applied Statistics*, vol. 22, no. 2, pp. 203–214, 1995.

[17] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin, "A modified principal component technique based on the lasso," *Journal of Computational and Graphical Statistics*, vol. 12, no. 3, pp. 531–547, 2003.

[18] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. Lanckriet, "A direct formulation for sparse pca using semidefinite programming," *SIAM review*, vol. 49, no. 3, pp. 434–448, 2007.
[19] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, 2006.
[20] H. Shen and J. Z. Huang, "Sparse principal component analysis via regularized low rank matrix approximation," *Journal of Multivariate Analysis*, vol. 99, no. 6, pp. 1015–1034, 2008.
[21] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, p. kxp008, 2009.
[22] R. Jenatton, G. Obozinski, and F. Bach, "Structured sparse principal component analysis," *arXiv preprint arXiv:0909.1440*, 2009.
[23] E. Ragnoli, S. McLoone, S. Lynn, J. Ringwood, and N. Macgearailt, "Identifying key process characteristics and predicting etch rate from high-dimension datasets," in *Advanced Semiconductor Manufacturing Conference, 2009. ASMC '09. IEEE/SEMI*, May 2009, pp. 106–111.
[24] P. Prakash, A. Johnston, B. Honari, and S. McLoone, "Optimal wafer site selection using forward selection component analysis," in *Advanced Semiconductor Manufacturing Conference (ASMC), 2012 23rd Annual SEMI*. IEEE, 2012, pp. 91–96.
[25] K. Li, J.-X. Peng, and E.-W. Bai, "A two-stage algorithm for identification of nonlinear dynamic systems," *Automatica*, vol. 42, no. 7, pp. 1189–1197, 2006.
[26] K. Li, J.-X. Peng, and Bai, "Two-stage mixed discrete–continuous identification of radial basis function (rbf) neural models for nonlinear systems," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 56, no. 3, pp. 630–643, 2009.
[27] I. T. Jolliffe, "Discarding variables in a principal component analysis. i: Artificial data," *Applied statistics*, pp. 160–173, 1972.
[28] W. Krzanowski, "Selection of variables to preserve multivariate data structure, using principal components," *Applied Statistics*, pp. 22–33, 1987.
[29] K. Mao, "Identifying critical variables of principal components for unsupervised feature selection," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 2, pp. 339–344, 2005.
[30] Y. Lu, I. Cohen, X. S. Zhou, and Q. Tian, "Feature selection using principal feature analysis," in *Proceedings of the 15th international conference on Multimedia*. ACM, 2007, pp. 301–304.
[31] Y. Cui and J. G. Dy, "Orthogonal principal feature selection," in *Sparse Optimization and Variable Selection Workshop at the International Conference on Machine Learning*. Helsinki, Finland, July 2008.
[32] M. Masaeli, Y. Yan, Y. Cui, G. Fung, and J. G. Dy, "Convex principal feature selection." in *SDM*. SIAM, 2010, pp. 619–628.
[33] D. C. Whitley, M. G. Ford, and D. J. Livingstone, "Unsupervised forward selection: a method for eliminating redundant variables," *Journal of Chemical Information and Computer Sciences*, vol. 40, no. 5, pp. 1160–1168, 2000.
[34] H.-L. Wei and S. A. Billings, "Feature subset selection and ranking for data dimensionality reduction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 1, pp. 162–166, 2007.
[35] R. Liu, R. Rallo, and Y. Cohen, "Unsupervised feature selection using incremental least squares," *International Journal of Information Technology and Decision Making*, vol. 10, no. 06, pp. 967–987, 2011.
[36] Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle, "Massively parallel feature selection: an approach based on variance preservation," *Machine Learning*, vol. 92, no. 1, pp. 195–220, 2013.
[37] V. Cevher and A. Krause, "Greedy dictionary selection for sparse representation," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 5, pp. 979–988, 2011.
[38] T. Zhang, "Adaptive forward-backward greedy algorithm for learning sparse representations," *Information Theory, IEEE Transactions on*, vol. 57, no. 7, pp. 4689–4708, 2011.
[39] P. Jain, A. Tewari, and I. S. Dhillon, "Orthogonal matching pursuit with replacement," in *Advances in Neural Information Processing Systems*, 2011, pp. 1215–1223.
[40] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 427–435.
[41] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.
[42] N. Rao, P. Shah, and S. Wright, "Forwardbackward greedy algorithms for signal demixing," in *Signals, Systems and Computers, 2014 48th Asilomar Conference on*. IEEE, 2014, pp. 437–441.
[43] H. Wold, "Nonlinear estimation by iterative least square procedures," in *Research Papers in Statistics*, F. David, Ed. Wiley, New York, 1966, pp. 411–444.
[44] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.
[45] M. S. Bartlett, "An inverse matrix adjustment arising in discriminant analysis," *Ann. Math. Statist*, vol. 22, no. 1, pp. 107–111, 03 1951.
[46] J. Jeffers, "Two case studies in the application of principal component analysis," *Applied Statistics*, pp. 225–236, 1967.
[47] H. Yue, S. Qin, R. Markle, C. Nauert, and M. Gatto, "Fault detection of plasma etchers using optical emission spectra," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 13, no. 3, pp. 374–385, Aug 2000.
[48] D. Zeng and C. Spanos, "Virtual metrology modeling for plasma etch operations," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 22, no. 4, pp. 419–431, Nov 2009.
[49] L. Puggini and S. McLoone, "Extreme learning machines for virtual metrology and etch rate prediction," in *Signals and Systems Conference (ISSC), 2015 26th Irish*. IEEE, 2015, pp. 1–6.

Seán McLoone received an M.E. degree in Electrical and Electronic Engineering and a PhD in Control Engineering from Queens University Belfast (QUB), Belfast, U.K. in 1992 and 1996, respectively. Following appointments as a Postdoctoral Research Fellow (1996-1997) and Lecturer at QUB (1998-2002) he joined the Department of Electronic Engineering at the National University of Ireland Maynooth in 2002, where he served as Senior Lecturer (2005-2012) and Head of Department (2009-2012). He is currently a Professor and Director of the Energy Power and Intelligent Control (EPIC) Research Cluster at Queens University Belfast. His research interests include computational intelligence techniques, data analytics, system identification and control, with a particular focus on smart grid and advanced manufacturing informatics applications.

Luca Puggini was born in Rome (Italy) in 1989. He obtained the Laurea Magistrale in Pure and Applied Mathematics from the University of Tor Vergata in 2013. He worked on his thesis at Statistics for Innovation in Oslo, Norway. In September 2013 he joined the Department of Electronic Engineering at Maynooth University as a PhD student on a collaborative research project with Intel Ireland. His research interests include statistics, big data, machine learning, and computational intelligence techniques.