SpaSM: A MATLAB Toolbox for Sparse Statistical Modeling

Sjöstrand, Karl; Clemmensen, Line Harder; Larsen, Rasmus; Einarsson, Gudmundur; Ersbøll, Bjarne Kjær

Published in:
Journal of Statistical Software

Link to article, DOI:
10.18637/jss.v084.i10

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Journal of Statistical Software

April 2018, Volume 84, Issue 10. doi: 10.18637/jss.v084.i10

SpaSM: A MATLAB Toolbox for Sparse Statistical Modeling

Karl Sjöstrand
EXINI Diagnostics AB

Line Harder Clemmensen
Technical University of Denmark

Gudmundur Einarsson
Technical University of Denmark

Rasmus Larsen
Technical University of Denmark

Bjarne Ersbøll
Technical University of Denmark

Abstract

Applications in biotechnology such as gene expression analysis and image processing have led to a tremendous development of statistical methods with emphasis on reliable solutions to severely underdetermined systems. Furthermore, interpretations of such solutions are of importance, meaning that the surplus of inputs has been reduced to a concise model. At the core of this development are methods which augment the standard linear models for regression, classification and decomposition such that sparse solutions are obtained. This toolbox aims at making public available carefully implemented and well-tested variants of the most popular of such methods for the MATLAB programming environment. These methods consist of easy-to-read yet efficient implementations of various coefficient-path following algorithms and implementations of sparse principal component analysis and sparse discriminant analysis which are not available in MATLAB. The toolbox builds on code made public in 2005 and which has since been used in several studies.

Keywords: least angle regression, LASSO, elastic net, sparse principal component analysis, sparse discriminant analysis, MATLAB.

1. Introduction

The introduction of the least angle regression (LAR) method for regularized/sparse regression (Efron, Hastie, Johnstone, and Tibshirani 2004) marked the starting point of a series of important contributions to the statistical computing community with the following common properties:

- Solutions are obtained sequentially along a path of gradually changing amounts of regularization. This paper focuses on methods where the method coefficients are piecewise linear functions of the regularization parameter, and where algorithms proceed by finding the next piecewise linear breakpoint.
- For sufficient amounts of l_1 regularization, solutions are *sparse*, i.e., some of the coefficients of the model are exactly zero, leading to more compact models which are easier to interpret.
- Methods are *efficient*, meaning they perform on a par with competing statistical methods when performance is measured on a test data set.

Examples of contributions are Zou and Hastie (2005), Zou, Hastie, and Tibshirani (2006), Rosset and Zhu (2007), Hastie, Rosset, Tibshirani, and Zhu (2004), Park and Hastie (2007), Friedman, Hastie, and Tibshirani (2010), of which the first three are detailed in this paper. The methods cover regression (the LASSO and the elastic net with ridge regression as a special case), classification (sparse discriminant analysis (SDA) with penalized linear discriminant analysis as a special case), and unsupervised modeling (sparse principal component analysis, SPCA). The goal of this paper is to provide reference MATLAB (The MathWorks Inc. 2017) implementations of these basic regularization-path oriented methods.

Currently there are no built-in implementations of least angle regression, SPCA or SDA in MATLAB. MATLAB includes an implementation of the LASSO and elastic net in the **Statistics and Machine Learning Toolbox**, but both are based on coordinate descent optimization instead of coordinate path tracking which is at the heart of the least angle regression method. The R package **glmnet** (Friedman *et al.* 2010) provides methods to work with generalized linear models via penalized maximum likelihood. The **glmnet** (Qian, Hastie, Friedman, Tibshirani, and Simon 2014) package is also available in MATLAB. This includes the LASSO and the elastic net, but both in R and MATLAB the optimization is done via coordinate descent. An implementation of least angle regression is available in the R package **lars** (Hastie and Efron 2013). There exist several isolated implementations of least angle regression online. There is one such implementation on the MATLAB Central File Exchange (Kim 2009).

There exist various problem formulations of SPCA. There exist several stand-alone implementations online. One such implementation, containing 8 problem formulations, is provided alongside (Richtárik, Takáč, and Ahipaşaoğlu 2012). An isolated MATLAB implementation can be found on the MATLAB Central File Exchange (Alsahaf 2015).

For SDA there also exist various problem formulations. Some of these options are implemented in R (Mai, Yang, and Zou in press, 2015; Witten 2015; Witten and Tibshirani 2011). Note that the same implementation as in the **SpaSM** toolbox is also available in the R package **sparseLDA** (Clemmensen and Kuhn 2016).

Currently the **SpaSM** toolbox is the only comprehensive toolbox for MATLAB that provides a variety of sparse methods based on least angle regression. We also present previously unpublished developments of the algorithm for sparse principal component analysis, and provide some evidence that performance is only slightly lowered (in terms of variance explained), while the computational complexity is significantly lowered. The implementation strikes a balance between performance and readability, making this toolbox a good starting point for learning the details of the methods. For this reason, the code is written as pure MATLAB scripts which closely follow the algorithms provided here. All methods have been fully described and

validated in their respective publications; despite this we provide terse but relatively complete derivations of each algorithm such that the paper can be read and the algorithms understood without having all references at hand.

The rest of the paper is structured as follows. Section 3 gives a short overview of the methods and files presented in the toolbox. Section 4 gives a concise derivation of the methods and pseudo code for the algorithms is provided. Section 5 is a short tutorial on how to apply the functions from the toolbox, interpret the command line output and a description of the input/output to the functions. The examples are shown on simulated data sets and all the examples can be found in the toolbox, with the appropriate seeds to generate the simulated data sets. Section 6 shows methods from the toolbox used on two real world data sets, one regarding Diabetes data and the second on shape data from human silhouettes.

2. Related work

Here we summarize some of the recent advances in sparse methods that are not included in the package. We point the reader to relevant software available and how it can be accessed from MATLAB.

Sparse regression. The Dantzig selector by Candes and Tao (2007) is similar to the LASSO in the sense that it performs regression and model selection. The main difference from the LASSO is in the way the optimization problem is formulated. The l_1 norm of the regression coefficients is minimized under constrains on the residuals. This can be formulated as a linear program. The main results also include bounds on the errors of the regression coefficients that are non-asymptotic. An implementation of the Dantzig selector can be found in the R package **flare** (Li, Zhao, Wang, Yuan, and Liu 2014) in the function `slim`.

Zou (2006) derives the necessary conditions for the LASSO variable selection to be consistent. He then proposes a new version of the LASSO called the adaptive LASSO. The modification of the traditional LASSO consists of adding weights to the regression coefficients in the penalty term, corresponding to the inverse of the OLS solution. An additional parameter γ is also added as the exponent of the weights for further tuning. It is proved that this method is consistent in variable selection and has the oracle-property, meaning that it performs as well as if the true underlying model was given in advance. This approach only works in the $p < n$ case, but one can use the ridge solution instead of the OLS solution to provide weights for the penalty term in the case of collinearity or the $p > n$ case. The drawback is that cross-validation must be performed over two parameters in the $p < n$ case, and three parameters in the $p > n$ case. This method is implemented in the R package **parcor** (Kraemer, Schaefer, and Boulesteix 2009) in the function `adalasso`.

Scaled sparse linear regression by Sun and Zhang (2012) gives a general approach to regression and penalization, with special focus on the LASSO. The idea is to estimate the parameters in the model and the noise level. By estimating the noise level, the penalization parameter can be adjusted in successive iterations of the algorithm. The parameter controlling the penalization thus becomes data dependent. Oracle inequalities are proved for prediction, estimation of noise level and regression coefficients. An implementation can be found in the R package **scalreg** (Sun 2013), the method can be used with the function `scalreg`.

For generalized linear models one can apply an l_1 norm regularizer via the R packages **glmnet**

(Friedman *et al.* 2010) and **penalized** (Goeman 2010). The main difference between the packages is the optimization of the likelihood functions. The **glmnet** package uses cyclical coordinate descent, while the **penalized** package uses a combination of gradient ascent and the Newton-Raphson algorithm. The **glmnet** package is also available for MATLAB (Qian *et al.* 2014).

A couple of R packages are available for general non-convex penalty functions. The **plus** package (Zhang 2010) has two main components for its MC_+ algorithm, namely a minimax concave penalty and penalized linear unbiased selection, which provides unbiased estimates of parameters and variable selection. The **ncvreg** package (Breheny and Huang 2011) also handles non-convex penalty functions but the optimization is done with coordinate descent.

Sparse graphical models. Sparse graphical models concern the estimation of edge weights in a graph where the nodes correspond to variables. Variables are connected in the graph if they are conditionally dependent. To achieve this one estimates a sparse inverse covariance matrix of a multivariate normal distribution from data. If an entry in the inverse covariance matrix is non-zero, then the corresponding variables are conditionally dependent. One of the variables could be a response variable, and thus these models can be used to model conditional dependence of the response to the predictors, like in traditional linear models.

In Meinshausen and Bühlmann (2006), the authors present a method to estimate the sparse inverse covariance matrix via neighborhood selection with the LASSO. They build the covariance matrix by using the LASSO on each variable separately. They show promising computational results. They also show that they get consistent estimation of the edges in the graph by controlling the probability of falsely joining some distinct connectivity components of the graph. An implementation is available in the R package **spaceExt** (He 2011) in the function `glasso.miss`.

Friedman, Hastie, and Tibshirani (2008) estimate the sparse inverse covariance matrix by starting with a blockwise coordinate descent approach and then solve the exact problem with a LASSO penalty using coordinate descent, instead of solving for each variable independently like Meinshausen and Bühlmann (2006). The method is implemented in the R package **glasso** (Friedman, Hastie, and Tibshirani 2014) via the function `glasso`.

Cai, Liu, and Luo (2011) propose a method (CLIME) where they use constrained l_1 minimization to estimate the sparse covariance matrix. They also show some generic results on the rate of convergence for different types of tails of population distributions. The problem can be solved with linear programming. An implementation is available in the R package **fastclime** (Pang, Qi, Liu, and Vanderbei 2016) via the function `fastclime.selector`.

Sparse quadratic discriminant analysis. Le and Hastie (2014) present a class of rules spanning from QDA to naïve Bayes through a path of sparse graphical models. The authors use a group LASSO penalty, which imposes sparsity on the same elements in all the K within class precision matrices, where K is the number of classes. The authors claim that the estimates of interactions from their method are easier to interpret than other classifiers that regularize QDA. The authors do not provide software.

Other implementations in MATLAB. Liu, Ji, Ye *et al.* (2009) present a MATLAB package called **SLEP** (sparse learning with efficient projections). They provide some MATLAB

implementations for efficient computation of lasso variants using optimization based on efficient Euclidean projections.

2.1. Calling R code from MATLAB

For comparisons, or to complement the methods in this toolbox, the R software packages referenced in the previous section can be run from within the MATLAB environment. There are a few ways to achieve this.

The most straight-forward approach is to write a separate R script and run it in batch mode with a system command in MATLAB. One way to achieve this from MATLAB would be:

```
>> system('R CMD BATCH rScript output');
```

The output from the R script (`rScript`) is written in the file `output`. The user then needs to load the results into MATLAB.

More generic approaches are also available, like the **MATLAB R-link** package available on the MATLAB Central File Exchange (Henson 2013). This solution only works on the Windows operating system and allows one to copy data back and forth from MATLAB and R.

3. In the toolbox

The toolbox consists of a series of MATLAB (The MathWorks Inc. 2017) scripts and functions to build and apply various statistical models for both supervised and unsupervised analyses. Below are listings of each method, file, subfunction and utility.

3.1. Methods

Forward selection: A variant of stepwise regression in which variables are included one-by-one based on their correlation with the current residual vector. Provides a baseline algorithm for other sparse methods for regression in this toolbox.

Least angle regression: Provides a more gentle version of the classical approach of forward selection regression. The algorithm is the basis for all other methods in the toolbox. The method is also an interesting statistical method in its own right.

LASSO: This method adds l_1 (1-norm) regularization to ordinary least squares regression, yielding solutions which are sparse in terms of the regression coefficients. This may lead to efficient suppression of noise and aids in interpretation.

Elastic net: Combining the algorithmic ideas of least angle regression, the computational benefits of ridge regression and the tendency towards sparse solutions of the LASSO, this versatile method is applicable for many data sets, also when the number of predictor variables far exceeds the number of observations. The corresponding LARS-EN algorithm is used in the implementation of the following two algorithms.

Sparse principal component analysis: Principal component analysis is a powerful tool for compacting a data set and for recovering latent structures in data, but solutions are

difficult to interpret as they involve all the original predictor variables. Sparse principal component analysis approximates the behavior of regular principal component analysis but models each component as a linear combination of a subset of the original variables.

Sparse linear discriminant analysis: Linear discriminant analysis is a standard tool for classification of observations into one of two or more groups. Further, the data can be visualized along the obtained discriminative directions. As with principal component analysis, these directions are combinations of all predictor variables. Sparse discriminant analysis reduces this to a subset of variables which may improve performance as well interpretability.

3.2. Files

`forwardselection.m` A baseline algorithm for variable selection. Based on the algorithm in `lar.m`.

`lar.m` An implementation of the LARS algorithm for least angle regression described by Efron *et al.* (2004).

`lasso.m` The LASSO method of Tibshirani (1996), implemented using a combination of the algorithms of Efron *et al.* (2004) and Rosset and Zhu (2007).

`elasticnet.m` The elastic net algorithm of Zou and Hastie (2005), with elements from Rosset and Zhu (2007).

`spca.m` The sparse principal component algorithm based on the work by Zou *et al.* (2006), with modification described below.

`slda.m` The sparse discriminant analysis of Clemmensen, Hastie, Witten, and Ersbøll (2011).

3.3. Sub-functions and utilities

`larsen.m` The actual implementation of the elastic net algorithm. The functions `lasso.m`, `elasticnet.m`, `spca.m` and `slda.m` depend on this function; however it is not intended for direct use.

`chol_insert.m` Update of the Cholesky factorization of $\mathbf{X}^\top\mathbf{X} + \delta\mathbf{I}$. Used in `lar.m` and `larsen.m`.

`chol_delete.m` Downdate of the above Cholesky factorization. Used in `larsen.m`.

`center.m` Convenience function for centering (removing the mean observation) a data matrix or response vector.

`normalize.m` Convenience function for centering and normalizing a data matrix or response matrix such that variables have unit Euclidean length.

Algorithm 1 Forward selection.

1: Initialize the active set $\mathcal{A} = \emptyset$ and the inactive set $\mathcal{I} = \{1 \dots p\}$.
2: Initialize the coefficient vector $\beta^{(0)} = \mathbf{0}$.
3: **for** $k \in \{0 \dots p-1\}$ **do**
4: Find the variable which is maximally correlated with the current residual $i = \arg\max_{i \in \mathcal{I}} \mathbf{x}_i^\top (\mathbf{y} - \mathbf{X}\beta^{(k)})$.
5: Move i from $\mathcal{I}$ to $\mathcal{A}$.
6: Update the active set coefficients $\beta_{\mathcal{A}}^{(k+1)} = (\mathbf{X}_{\mathcal{A}}^\top \mathbf{X}_{\mathcal{A}})^{-1} \mathbf{X}_{\mathcal{A}}^\top \mathbf{y}$.
7: **end for**
8: Output the series of coefficients $\mathbf{B} = [\beta^{(0)} \dots \beta^{(p)}]$.

4. Methods and algorithms

This section presents the principles behind each method in the toolbox, and outlines their algorithms. The basic building block is the LARS-EN algorithm (Zou and Hastie 2005) which encompasses regression via ordinary least squares, ridge regression, the LASSO and the elastic net. These are based on the linear model $\mathbf{y} = \mathbf{X}\beta + \boldsymbol{\varepsilon}$ where $\mathbf{y}$ $(n \times 1)$ is the observed response variable, $\mathbf{X}$ $(n \times p)$ is the data matrix where the ith column represents the ith predictor variable, β $(p \times 1)$ is the set of model coefficients which determines the load on each predictor variable, and $\boldsymbol{\varepsilon}$ are the residual errors. Unless stated otherwise, $\mathbf{y}$ is assumed centered and $\mathbf{X}$ is assumed centered and normalized such that each variable has zero mean and unit Euclidean length. A sparse method for regression estimates a coefficient vector β with many zero elements, giving an estimate $\hat{\mathbf{y}}$ of $\mathbf{y}$ which is a linear combination of a subset of available variables in $\mathbf{X}$. Sparse solutions may be preferred to full counterparts if the latent linear model can be assumed to be sparse, or when interpretation of the results is important. The set $\mathcal{A}$ denotes the indices in β corresponding to non-zero elements; we refer to this as the *active set*. The set $\mathcal{I}$ is called the *inactive set* and denotes the complement of $\mathcal{A}$. We use these sets also to denote submatrices such as the $(n \times |\mathcal{A}|)$ matrix $\mathbf{X}_{\mathcal{A}}$, consisting of the columns (variables) of $\mathbf{X}$ corresponding to the indices in $\mathcal{A}$. All algorithms proceed in iterations and we indicate iteration number by a parenthesized superscript number, e.g., $\hat{\beta}^{(k)}$ for the regression coefficients calculated in the kth iteration. It is further convenient to define an operator $\min^+(\cdot)$ which finds the smallest strictly positive value of the (vector-valued) input.

The methods for regression described below proceed in an iterative manner, adding or subtracting variables in the model in each step. The methods start with the trivial constant model, then move towards the full representation which corresponds to ordinary least squares regression or ridge regression, depending on the type of regularization. To put the presentation of these algorithms into perspective, we begin with a quick review of one of the simplest algorithms of this kind.

In *forward selection*, a variant of *stepwise regression*, variables are added one-by-one until some goodness-of-fit criterion is fulfilled. The next variable to include in this scheme can be chosen based on a number of criteria. The methods in this toolbox generally pick the variable that has the highest absolute correlation with the current residual vector. To fix the terminology and to give a simple baseline algorithm we state a forward selection algorithm in Algorithm 1.

In this algorithm, we move to the least squares solution using all currently active variables in each step. This approach is known as a greedy method. The following sections cover less greedy variations on the forward selection scheme which result in algorithms with generally better performance and which are able to handle more difficult data sets.

4.1. Least angle regression

Least angle regression (LAR) is a regression method that provides a more gentle version of forward selection. Conceptually, LAR modifies Algorithm 1 on only one account. Instead of choosing a step size which yields the (partial) least squares solution in each step, we shorten the step length such that we stop when any inactive variable becomes equally important as the active variables in terms of correlation with the residual vector. That variable is then included in the active set and a new direction is calculated. Recall that all active variables are uncorrelated with the residual vector at the least squares solution, the step length will therefore always be as short or shorter at the point where we find the next active variable to include than that of the least squares solution.

The algorithm starts with the empty set of active variables. The correlation between each variable and the response is measured, and the variable with the highest correlation becomes the first variable included into the model. The first direction is then towards the least squares solution using this single active variable. Walking along this direction, the angles between the variables and the residual vector are measured. Along this walk, the angles will change; in particular, the correlation between the residual vector and the active variable will shrink linearly towards 0. At some stage before this point, another variable will obtain the same correlation with respect to the residual vector as the active variable. The walk stops and the new variable is added to the active set. The new direction of the walk is towards the least squares solution of the two active variables, and so on. After p steps, the full least squares solution will be reached.

The LAR algorithm is efficient since there is a closed form solution for the step length at each stage. Denoting the model estimate of $\mathbf{y}$ at iteration k by $\hat{\mathbf{y}}^{(k)}$ and the least squares solution including the newly added active variable $\hat{\mathbf{y}}_{\text{OLS}}^{(k+1)}$, the walk from $\hat{\mathbf{y}}^{(k)}$ towards $\hat{\mathbf{y}}_{\text{OLS}}^{(k+1)}$ can be formulated $(1-\gamma)\hat{\mathbf{y}}^{(k)} + \gamma\hat{\mathbf{y}}_{\text{OLS}}^{(k+1)}$ where $0 \leq \gamma \leq 1$. Estimating $\hat{\mathbf{y}}^{(k+1)}$, the position where the next active variable is to be added, then amounts to estimating γ. We seek the smallest positive γ where correlations become equal, that is

$$\mathbf{x}_{i\in\mathcal{I}}^\top(\mathbf{y} - (1-\gamma)\hat{\mathbf{y}}^{(k)} - \gamma\hat{\mathbf{y}}_{\text{OLS}}^{(k+1)}) = \mathbf{x}_{j\in\mathcal{A}}^\top(\mathbf{y} - (1-\gamma)\hat{\mathbf{y}}^{(k)} - \gamma\hat{\mathbf{y}}_{\text{OLS}}^{(k+1)}).$$

Solving this expression for γ, we get

$$\gamma = \frac{(\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{y} - \hat{\mathbf{y}}^{(k)})}{(\mathbf{x}_i - \mathbf{x}_j)^\top(\hat{\mathbf{y}}_{\text{OLS}}^{(k+1)} - \hat{\mathbf{y}}^{(k)})} = \frac{(\mathbf{x}_i - \mathbf{x}_j)^\top\boldsymbol{\varepsilon}}{(\mathbf{x}_i - \mathbf{x}_j)^\top\mathbf{d}},$$

where $\mathbf{d} = \hat{\mathbf{y}}_{\text{OLS}}^{(k+1)} - \hat{\mathbf{y}}^{(k)}$ is the direction of the walk, and $j \in \mathcal{A}$. Now, $\mathbf{d}$ is the orthogonal projection of $\boldsymbol{\varepsilon}$ onto the plane spanned by the variables in $\mathcal{A}$, therefore we have $\mathbf{x}_j^\top\boldsymbol{\varepsilon} = \mathbf{x}_j^\top\mathbf{d} \equiv c$, representing the angle at the current breakpoint $\hat{\mathbf{y}}^{(k)}$. Furthermore, the sign of the correlation between variables is irrelevant. Therefore, we have

$$\gamma = \min_{i\in\mathcal{I}}{}^{+}\left\{\frac{\mathbf{x}_i^\top\boldsymbol{\varepsilon} - c}{\mathbf{x}_i^\top\mathbf{d} - c}, \quad \frac{\mathbf{x}_i^\top\boldsymbol{\varepsilon} + c}{\mathbf{x}_i^\top\mathbf{d} + c}\right\}, \quad 0 < \gamma \leq 1,$$

Algorithm 2 Least angle regression.

Initialize the coefficient vector $\beta^{(0)} = \mathbf{0}$ and the fitted vector $\hat{\mathbf{y}}^{(0)} = \mathbf{0}$,
Initialize the active set $\mathcal{A} = \emptyset$ and the inactive set $\mathcal{I} = \{1 \dots p\}$.
for $k = 0$ **to** $p - 2$ **do**
  Update the residual $\boldsymbol{\varepsilon} = \mathbf{y} - \hat{\mathbf{y}}^{(k)}$.
  Find the maximal correlation $c = \max_{i \in \mathcal{I}} |\mathbf{x}_i^\top \boldsymbol{\varepsilon}|$.
  Move variable corresponding to c from $\mathcal{I}$ to $\mathcal{A}$.
  Calculate the least squares solution $\beta_{\text{OLS}}^{(k+1)} = (\mathbf{X}_\mathcal{A}^\top \mathbf{X}_\mathcal{A})^{-1} \mathbf{X}_\mathcal{A}^\top \mathbf{y}$.
  Calculate the current direction $\mathbf{d} = \mathbf{X}_\mathcal{A} \beta_{\text{OLS}}^{(k+1)} - \hat{\mathbf{y}}^{(k)}$.
  Calculate the step length $\gamma = \min_{i \in \mathcal{I}}^{+} \left\{ \frac{\mathbf{x}_i^\top \boldsymbol{\varepsilon} - c}{\mathbf{x}_i^\top \mathbf{d} - c}, \quad \frac{\mathbf{x}_i^\top \boldsymbol{\varepsilon} + c}{\mathbf{x}_i^\top \mathbf{d} + c} \right\}$, $0 < \gamma \leq 1$.
  Update regression coefficients $\beta^{(k+1)} = (1 - \gamma)\beta^{(k)} + \gamma \beta_{\text{OLS}}^{(k+1)}$.
  Update the fitted vector $\hat{\mathbf{y}}^{(k+1)} = \hat{\mathbf{y}}^{(k)} + \gamma \mathbf{d}$.
end for
Let $\beta^{(p)}$ be the full least squares solution $\beta^{(p)} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$.
Output the series of coefficients $\mathbf{B} = [\beta^{(0)} \dots \beta^{(p)}]$.

where the two terms are for correlations/angles of equal and opposite sign respectively. The coefficients at this next step are given by

$$\beta^{(k+1)} = (1 - \gamma)\beta^{(k)} + \gamma \beta_{\text{OLS}}^{(k+1)}.$$

Given these key pieces of the LAR algorithm, we state the entire procedure in Algorithm 2.

Each step of Algorithm 2 adds a covariate to the model until the full least squares solution is reached. It is natural to parameterize this process by the size $s(\beta)$ of the coefficients at each step as well as in-between steps of the algorithm. The algorithm returns the following parametrization,

$$s(\beta) = \|\beta\|_1 = \sum_{i=1}^{p} |\beta_i|. \tag{1}$$

Picking a suitable model for a particular analysis thus means selecting a suitable value of $s(\beta) \in (0, \|\beta_{\text{OLS}}\|_1)$. Cross-validation or an independent validation data set are obvious choices for this purpose, however, the algorithm provides information which substitutes or complements this process.

Degrees of freedom: Efron *et al.* (2004) showed that the number of degrees of freedom at each step of the LAR algorithm is well approximated by the number of non-zero elements of β. The algorithm therefore returns the following sequence,

$$df_{\text{LAR}}^{(k)} = |\mathcal{A}| = k, \quad k = 0 \dots p.$$

Mallow's C_p: Given the above measure of the number of degrees of freedom, we can calculate a number of model selection criteria. Mallow's C_p measure is defined as (Zou, Hastie, and Tibshirani 2007)

$$C_p^{(k)} = \frac{1}{\sigma_\varepsilon^2} \|\mathbf{y} - \mathbf{X}\beta^{(k)}\|^2 - n + 2df^{(k)}.$$

Akaike's information criterion: Akaike's information criterion is similar to Mallow's C_p and is defined as

$$AIC^{(k)} = \|\mathbf{y} - \mathbf{X}\beta^{(k)}\|^2 + 2\sigma_\varepsilon^2 df^{(k)}.$$

Bayesian information criterion: The Bayesian information criterion tends to choose a sparser model than both AIC and C_p and is defined as

$$BIC^{(k)} = \|\mathbf{y} - \mathbf{X}\beta^{(k)}\|^2 + \log(n)\sigma_\varepsilon^2 df^{(k)}.$$

The latter three criteria can be used to pick a suitable model, typically indicated by the smallest value of each criterion. Alternatively, one can choose the sparsest model for which more complex models lead to scant improvements in the relevant model selection criterion. The measure σ_ε^2 represents the residual variance of a low-bias model which is here defined as

$$\sigma_\varepsilon^2 = \frac{1}{n}\|\mathbf{y} - \mathbf{X}^\dagger\mathbf{y}\|^2,$$

where $\mathbf{X}^\dagger$ is the Moore-Penrose pseudo-inverse of $\mathbf{X}$, equivalent to a ridge regression solution $\arg\min \|\mathbf{y}-\mathbf{X}\beta\|^2+\lambda\|\beta\|^2$ in the limit $\lambda \to 0$. Note that in cases where $p > n$, this measure of the residual variance will be zero which in effect turns the information criteria defined above into a measure of training error only. We therefore recommend using these criteria for model selection only in cases where n is well above p.

The key computational burden of Algorithm 2 lies in Step 7 where the OLS solution involving the variables in $\mathcal{A}$ is calculated. Two techniques are used to alleviate this. For problems where n is at least ten times larger than p, we calculate the full Gram matrix $\mathbf{X}^\top\mathbf{X}$ once and use the submatrix $\mathbf{X}_\mathcal{A}^\top\mathbf{X}_\mathcal{A}$ to find $\beta_{\text{OLS}}^{(k+1)}$, thus avoiding an $\mathcal{O}(|\mathcal{A}|^2 n)$ matrix multiplication. When $p > 1000$, this method is not preferred since the memory footprint of the resulting $p \times p$ Gram matrix may pose a problem. In cases where $10n < p$, or when a pre-computed Gram matrix is impractical, we maintain a matrix $\mathbf{R}$ of the Cholesky factorization of the current Gram (sub)matrix $\mathbf{X}_\mathcal{A}^\top\mathbf{X}_\mathcal{A}$ such that $\mathbf{R}^\top\mathbf{R} = \mathbf{X}_\mathcal{A}^\top\mathbf{X}_\mathcal{A}$. As variables join the active set, $\mathbf{R}$ can be updated with low computational cost. The conditions chosen for selecting between the two methods are not exact, but we have gathered evidence through simulation studies that they work well on most standard computers.

In practice one frequently has a notion of the sparsity of the desired solution when running Algorithm 2. To avoid unnecessary computations, the algorithm can be stopped prematurely, either when the active set reaches a certain size, or when the l_1 norm of the coefficients in $\beta^{(k)}$ exceeds a preset threshold. Optionally, the algorithm stores and returns the solution fulfilling the specified sparsity criterion only in order to save computer resources. This direct controlling of sparsity is a clear advantage over the coordinate descent method, where the number of non-zero parameters in the model cannot be specified directly.

4.2. The LASSO

The LASSO (Tibshirani 1996) represents the most basic augmentation of the ordinary least squares solution which implements coefficient shrinkage and selection. The sum of squared

residuals loss function $L(\hat{\beta}(\lambda))$ is combined with a penalty function $J(\hat{\beta}(\lambda))$ based on the l_1 norm as,

$$\hat{\beta}(\lambda) = \arg\min_{\beta} L(\hat{\beta}(\lambda)) + \lambda J(\hat{\beta}(\lambda)) = \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\|\beta\|_1. \tag{2}$$

The l_1 penalty will promote sparse solutions. This means that as λ is increased, elements of $\hat{\beta}(\lambda)$ will become exactly zero. Due to the non-differentiability of the penalty function, there are no closed-form solutions to Equation 2. A number of algorithms have been proposed (e.g., Fu 1998; Osborne, Presnell, and Turlach 2000; Friedman *et al.* 2010) including the quadratic programming approach on an expanded space of variables outlined in the original LASSO paper of Tibshirani (1996). The algorithm presented here is due to Rosset and Zhu (2007) who derived a sufficient condition for piecewise linear coefficient paths on which they based several LASSO-type methods. The LASSO algorithm described here is a special case of their work. Efron *et al.* (2004) arrived at an equivalent algorithm by showing that a small modification to the least angle algorithm yields LASSO solutions.

The goal of this section is to derive an expression for how the solutions of Equation 2 change with λ. The solution set $\hat{\beta}(\lambda)$ will hit a non-differentiability point when coefficients either go from non-zero to zero (join $\mathcal{I}$), or the other way around (join $\mathcal{A}$). Assume first that we are in a region of values of λ where variables are neither joining nor leaving $\mathcal{A}$. The normal equations to Equation 2 around λ and around a nearby point $\lambda + \epsilon$ are then

$$-2\mathbf{X}_{\mathcal{A}}^{\top}(\mathbf{y} - \mathbf{X}_{\mathcal{A}}\hat{\beta}_{\mathcal{A}}(\lambda)) + \lambda \cdot \text{sign}(\hat{\beta}_{\mathcal{A}}(\lambda)) = 0 \tag{3}$$

$$-2\mathbf{X}_{\mathcal{A}}^{\top}(\mathbf{y} - \mathbf{X}_{\mathcal{A}}\hat{\beta}_{\mathcal{A}}(\lambda + \epsilon)) + (\lambda + \epsilon) \cdot \text{sign}(\hat{\beta}_{\mathcal{A}}(\lambda + \epsilon)) = 0. \tag{4}$$

We now write Equation 4 as a first order Taylor expansion around $\hat{\beta}(\lambda)$. The general form of a multivariate Taylor expansion of $\mathbf{f}(x)$ around a is

$$\mathbf{f}(x) = \sum_{k=0}^{\infty} \frac{\nabla^{(k)}\mathbf{f}(a)}{k!}(x-a)^k = \mathbf{f}(a) + \nabla\mathbf{f}(a)(x-a) + \frac{1}{2}\nabla^2\mathbf{f}(a)(x-a)^2 + \ldots.$$

We have,

$$\begin{aligned}
\mathbf{f}(\hat{\beta}(\lambda)) &= -2\mathbf{X}_{\mathcal{A}}^{\top}(\mathbf{y} - \mathbf{X}_{\mathcal{A}}\hat{\beta}_{\mathcal{A}}(\lambda)) + (\lambda + \epsilon) \cdot \text{sign}(\hat{\beta}_{\mathcal{A}}(\lambda)) \\
\nabla\mathbf{f}(\hat{\beta}(\lambda)) &= 2\mathbf{X}_{\mathcal{A}}^{\top}\mathbf{X}_{\mathcal{A}} \\
\nabla^{(k)}\mathbf{f}(\hat{\beta}(\lambda)) &= 0 \quad \text{for } k = 2 \ldots \infty.
\end{aligned}$$

The complete expansion of Equation 4 becomes

$$-2\mathbf{X}_{\mathcal{A}}^{\top}(\mathbf{y} - \mathbf{X}_{\mathcal{A}}\hat{\beta}_{\mathcal{A}}(\lambda)) + (\lambda + \epsilon) \cdot \text{sign}(\hat{\beta}_{\mathcal{A}}(\lambda)) + 2\mathbf{X}_{\mathcal{A}}^{\top}\mathbf{X}_{\mathcal{A}}\left(\hat{\beta}_{\mathcal{A}}(\lambda + \epsilon) - \hat{\beta}_{\mathcal{A}}(\lambda)\right) = 0.$$

Using Equation 3, we can rearrange this expression to

$$\frac{\hat{\beta}_{\mathcal{A}}(\lambda + \epsilon) - \hat{\beta}_{\mathcal{A}}(\lambda)}{\epsilon} = -(2\mathbf{X}_{\mathcal{A}}^{\top}\mathbf{X}_{\mathcal{A}})^{-1}\text{sign}(\hat{\beta}_{\mathcal{A}}(\lambda)), \tag{5}$$

which approaches $\nabla\hat{\beta}(\lambda)$ as $\epsilon \to 0$ (using $\nabla\hat{\beta}_{\mathcal{I}}(\lambda) = 0$). This is a constant function which means that the coefficient paths between *events* (changes to $\mathcal{A}$ and $\mathcal{I}$) are piecewise linear, similarly to least angle regression.

Now that an expression for the change in $\hat{\beta}(\lambda)$ between events has been established, we focus on finding the values of λ for which changes to $\mathcal{A}$ and $\mathcal{I}$ take place. It is beneficial here to consider Equation 2 on an expanded set of β-values, chosen such that $\beta_j = \beta_j^+ + \beta_j^-$ where $\beta_j^+ \geq 0$ and $\beta_j^- \geq 0$, $\forall j$,

$$\arg\min_{\beta^+,\beta^-} \|\mathbf{y} - \mathbf{X}(\beta^+ - \beta^-)\|^2 + \lambda\|(\beta^+ + \beta^-)\|_1 = L(\hat{\beta}(\lambda)) + \lambda\|(\beta^+ + \beta^-)\|_1 \qquad (6)$$
$$\text{such that} \quad \beta_j^+ \geq 0, \beta_j^- \geq 0, \forall j.$$

This formulation of the LASSO is differentiable, at the price of having to deal with twice as many variables. The Lagrange primal function is

$$L(\hat{\beta}(\lambda)) + \lambda\|(\beta^+ + \beta^-)\|_1 - \sum_{j=1}^{p} \lambda_j^+ \beta_j^+ - \sum_{j=1}^{p} \lambda_j^- \beta_j^- , \quad \lambda_j^+ \geq 0, \lambda_j^- \geq 0, \forall j, \qquad (7)$$

where we have introduced the Lagrange multipliers λ_j^+ and λ_j^-. The Karush-Kuhn-Tucker conditions are

$$(\nabla L(\beta))_j + \lambda - \lambda_j^+ = 0 \qquad (8)$$
$$-(\nabla L(\beta))_j + \lambda - \lambda_j^- = 0 \qquad (9)$$
$$\lambda_j^+ \beta_j^+ = 0 \qquad (10)$$
$$\lambda_j^- \beta_j^- = 0. \qquad (11)$$

From these conditions, a number of useful properties arise. First, we note that setting $\lambda = 0$ indeed gives us (using Equations 8 and 9) $\nabla L(\beta) = 0$ as expected. For positive values of λ we have,

$$\beta_j^+ > 0+ \Rightarrow +\lambda_j^+ = 0 \Rightarrow \nabla L(\beta) = -\lambda \Rightarrow \lambda_j^- > 0 \Rightarrow \beta_j^- = 0 \qquad (12)$$
$$\beta_j^- > 0+ \Rightarrow +\lambda_j^- = 0 \Rightarrow \nabla L(\beta) = \lambda \Rightarrow \lambda_j^+ > 0 \Rightarrow \beta_j^+ = 0. \qquad (13)$$

Elements in $\mathcal{A}$ have either $\beta_j^+ > 0$ or $\beta_j^- > 0$, but cannot both be non-zero. That is,

$$|(\nabla L(\beta))_j| = \lambda, \quad j \in \mathcal{A} \qquad (14)$$
$$|(\nabla L(\beta))_j| \leq \lambda, \quad j \in \mathcal{I}.$$

We are seeking the value of $\gamma > 0$ for which a variable in $\mathcal{I}$ joins $\mathcal{A}$ or vice versa. We have arrived at the following conditions,

$$j \in \mathcal{A} \to \mathcal{I}: \quad \hat{\beta}_j^{(k)} + \gamma \nabla \hat{\beta}_j^{(k)} = 0, \quad j \in \mathcal{A} \qquad (15)$$
$$j \in \mathcal{I} \to \mathcal{A}: \quad |(\nabla L(\hat{\beta}^{(k)} + \gamma \nabla \hat{\beta}^{(k)}))_i| = |(\nabla L(\hat{\beta}^{(k)} + \gamma \nabla \hat{\beta}^{(k)}))_j|, \quad j \in \mathcal{A}, i \in \mathcal{I}. \qquad (16)$$

The first of these expressions defines the distances $\{\gamma\}$ at which active variables hit zero and join $\mathcal{I}$. The second expression defines the distances at which inactive variables violate the second condition in Equation 14 and thus must join $\mathcal{A}$. Note that any element in $\mathcal{A}$ can be chosen to calculate the RHS of Equation 16, they all equal λ. The smallest value $\gamma_{\min}$ of the distances $\{\gamma\}$ is where the next event will happen. The coefficients can now be updated by

$$\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} + \gamma_{\min} \nabla \hat{\beta}^{(k)}. \qquad (17)$$

Algorithm 3 LASSO (Rosset and Zhu 2007).

Initialize $\beta^{(0)} = \mathbf{0}$, $\mathcal{A} = \arg\max_j |\mathbf{x}_j^\top \mathbf{y}|$, $\nabla\hat{\beta}_{\mathcal{A}}^{(0)} = -\text{sign}(\mathbf{x}_{\mathcal{A}}^\top \mathbf{y})$, $\nabla\hat{\beta}_{\mathcal{I}}^{(0)} = 0$, $k = 0$.
while $\mathcal{I} \neq \emptyset$ **do**
  $\gamma_j = \min^+_{j\in\mathcal{A}} \; -\beta_j^{(k)} / \nabla\hat{\beta}_j^{(k)}$
  $\gamma_i = \min^+_{i\in\mathcal{I}} \; \{\frac{(\mathbf{x}_i+\mathbf{x}_j)^\top(\mathbf{y}-\mathbf{X}\hat{\beta}^{(k)})}{(\mathbf{x}_i+\mathbf{x}_j)^\top(\mathbf{X}\nabla\hat{\beta}^{(k)})}, \frac{(\mathbf{x}_i-\mathbf{x}_j)^\top(\mathbf{y}-\mathbf{X}\hat{\beta}^{(k)})}{(\mathbf{x}_i-\mathbf{x}_j)^\top(\mathbf{X}\nabla\hat{\beta}^{(k)})}\}$, where j is any index in $\mathcal{A}$
  $\gamma = \min\{\gamma_j, \gamma_i\}$
  if $\gamma = \gamma_j$ **then**
    Move j from $\mathcal{A}$ to $\mathcal{I}$.
  else
    Move i from $\mathcal{I}$ to $\mathcal{A}$.
  end if
  $\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} + \gamma\nabla\hat{\beta}^{(k)}$
  $\nabla\hat{\beta}_{\mathcal{A}}^{(k+1)} = -(2\mathbf{X}_{\mathcal{A}}^\top\mathbf{X}_{\mathcal{A}})^{-1} \cdot \text{sign}(\hat{\beta}_{\mathcal{A}}^{(k+1)})$
  $k = k + 1$
end while
Output the series of coefficients $\mathbf{B} = [\beta^{(0)} \ldots \beta^{(k)}]$.

We have arrived at Algorithm 3 for the LASSO.

One of the benefits with this particular algorithm is that the coefficient path can be parameterized either in terms of $\|\hat{\beta}^{(k)}\|_1$, the size of the penalty at iteration k, or the regularization parameter λ. The latter is seldom explicitly specified in path-following algorithms but here, the first identity in Equation 14 provides a way of directly calculating λ as a function of $\hat{\beta}$,

$$\lambda = 2|\mathbf{x}_{j\in\mathcal{A}}^\top(\mathbf{y} - \mathbf{X}\hat{\beta})|. \tag{18}$$

Any element in $\mathcal{A}$ will do for this calculation. To minimize the risk of numerical problems, we calculate this value for all elements in $\mathcal{A}$ and pick the median.

If asked for, the algorithm returns the same information as Algorithm 2. The LASSO solution path can be parameterized either in terms of $s(\beta)$ (cf., Equation 1), or in terms of λ which also can be interpreted as a function of β, cf., Equation 18. Zou *et al.* (2007) show that an unbiased estimate of the degrees of freedom of a particular LASSO solution is given by $|\mathcal{A}|$, the number of non-zero components of β. Given this estimate, the various model selection criteria can be calculated as outlined in Section 4.1.

We use the same Gram matrix or Cholesky updating scheme as described in Section 4.1. As variables leave the active set, the Cholesky factorization $\mathbf{R}^\top\mathbf{R} = \mathbf{X}_{\mathcal{A}}^\top\mathbf{X}_{\mathcal{A}}$ is downdated by removing the contribution to $\mathbf{R}$ which is due to the dropped variable.

4.3. The elastic net

Ridge regression (Hoerl and Kennard 1970) represents an effective way of shrinking the OLS coefficients towards zero. The l_1 penalty of the LASSO is replaced with an l_2 penalty,

$$\hat{\beta}(\delta) = \arg\min_\beta \|\mathbf{y} - \mathbf{X}\beta\|^2 + \delta\|\beta\|^2, \tag{19}$$

which leads to the closed form solution

$$\hat{\beta}(\delta) = (\mathbf{X}^\top \mathbf{X} + \delta \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}. \tag{20}$$

Although similar in formulation, ridge regression and the LASSO have important differences. The l_2 penalty of ridge regression leads to a shrinkage of the regression coefficients, much like the l_1 penalty of the LASSO, but coefficients are not forced to be exactly zero for finite values of δ. However, a benefit of ridge regression is that a unique solution is available, also when the data matrix $\mathbf{X}$ is rank deficient, e.g., when there are more predictors than observations ($p > n$). This is seen in Equation 20; the addition of a sufficiently large constant value along the diagonal of the Gram matrix $\mathbf{X}^\top \mathbf{X}$ ensures full rank (Petersen and Pedersen 2008). The LASSO algorithm (Algorithm 3) is terminated when the active set size $|\mathcal{A}|$ becomes larger than p since the matrix $\mathbf{X}_\mathcal{A}^\top \mathbf{X}_\mathcal{A}$ in Step 12 is no longer invertible.

Elastic net regression (Zou and Hastie 2005) combines the virtues of ridge regression and the LASSO by considering solutions penalized by both an l_2 and an l_1 term,

$$\hat{\beta}(\lambda, \delta) = \arg\min_\beta \|\mathbf{y} - \mathbf{X}\beta\|^2 + \delta\|\beta\|^2 + \lambda\|\beta\|_1, \tag{21}$$

thus bridging the gap between the LASSO ($\delta = 0$) and ridge regression ($\lambda = 0$). The l_2 penalty ensures a unique solution also when $p > n$ and the l_1 penalty offers variable selection via a sparse vector of coefficients $\hat{\beta}$. Moreover, the l_2 penalty leads to a *grouping effect* (Zou and Hastie 2005), a term that alludes to the characteristic that highly correlated predictors tend to have similar regression coefficients for nonzero δ. Note however that this does in general not mean that highly correlated variables are included into the active set in groups along the regularization path.

We can use the LASSO algorithm to obtain the full regularization path of elastic net solutions. To see this, we first note that ridge regression solutions can be obtained by solving an ordinary least squares problem with an augmented set of observations,

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\delta}\mathbf{I}_p \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}. \tag{22}$$

Expanding the equation $\hat{\beta} = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{y}}$ gives the ridge solution in Equation 20. For a fixed value of δ, Algorithm 3 offers solutions for all relevant values of λ. Selecting suitable values for the regularization parameters typically involves selecting the best value of λ for a discrete set of values of δ. Thus, the algorithm must be run for each value of δ.

If $p > n$, the augmented data matrix in Equation 22 has size $(n + p) \times p$, implying a system of equations that may be prohibitively large. Remarkably, it turns out that we can do without explicitly forming these augmented matrices, mainly due to the fact that any multiplications with $\tilde{\mathbf{y}}$ effectively voids the contribution of the additional rows in $\tilde{\mathbf{X}}$ since the corresponding rows of $\tilde{\mathbf{y}}$ are zero. Other computations are dot products between vectors with additional elements in $\mathcal{I}$ and vectors with additional elements in $\mathcal{A}$. Since these never coincide ($\mathcal{I} = \mathcal{A}^c$), these additional elements do not contribute to the result. The partial OLS solution calculated in Step 12 must however take into account the additional rows $\mathbf{X}$. When this equation is solved using a pre-computed Gram matrix, we simply supply the augmented Gram matrix $\mathbf{X}^\top \mathbf{X} + \delta \mathbf{I}$. When the Cholesky approach is used, it is straight-forward to take an additional parameter δ into account such that the Cholesky factorization of $\mathbf{X}^\top \mathbf{X} + \delta \mathbf{I}$ is obtained. Except for

these alterations to Step 12, the algorithm is run as usual. The LASSO and the elastic net therefore use the same underlying function (`larsen.m`) which takes the additional parameter δ used in Step 12. For LASSO solutions δ is simply set to zero.

Zou and Hastie (2005) argue and provide some evidence that the double shrinkage introduced by the l_1 and l_2 penalty has an unfortunate effect on prediction accuracy. They propose to compensate for this by multiplying the solutions $\mathbf{B}$ by a factor $(1 + \delta)$, and refer to the unadjusted solutions as the naïve elastic net. In some cases, the naïve solution is preferred, which consequently i obtained either by calling `larsen.m` directly or by dividing the elastic net solutions by $(1 + \delta)$.

The elastic net algorithm outputs the same model selection criteria as the LAR and LASSO algorithms. Computationally, the difference lies in the estimation of the number of degrees of freedom and the residual variance σ_ε^2. For non-zero δ, the corresponding ridge regression solution is used as a low-bias model in the estimation of the latter. Zou (2005) shows that an unbiased estimate of the number of degrees of freedom of elastic net solutions can be obtained by

$$\operatorname{tr}\left(\mathbf{X}_\mathcal{A}(\mathbf{X}_\mathcal{A}^\top\mathbf{X}_\mathcal{A} + \delta\mathbf{I})^{-1}\mathbf{X}_\mathcal{A}^\top\right),$$

which we solve efficiently using a singular value decomposition of $\mathbf{X}_\mathcal{A}$.

4.4. Sparse principal component analysis

Principal component analysis (PCA) is a linear transformation $\mathbf{S} = \mathbf{XL}$ of a mean-zero data matrix $\mathbf{X}$ where the *loading vectors* (columns) of $\mathbf{L}$ provide an orthonormal basis which successively maximizes the variance of the projected data in $\mathbf{S}$, where the *principal components* (columns) of $\mathbf{S}$ are uncorrelated, see, e.g., Hastie, Tibshirani, and Friedman (2009). PCA is optimal in the sense that no linear transformation can produce a more compact representation of data given $K < p$ basis vectors. The successive maximization of variance means that the first few principal components are usually sufficient to accurately describe the data. However, each principal component is a linear combination of *all* variables in $\mathbf{X}$ and is therefore difficult to interpret and assign a meaningful label. To alleviate this, sparse PCA (SPCA) aims at upholding some or all of the properties of PCA – successive maximization of variance, independence of the loading vectors and uncorrelated principal components – while enforcing sparsity of the loading vectors such that each principal component is a linear combination of only a few of the original variables.

The algorithm for computing sparse loading vectors used in this toolbox is detailed in Zou *et al.* (2006), and uses the elastic net in a regression-like framework for PCA. In the spirit of this paper, we start by formulating regular PCA as the solution to a regression problem, and then add suitable constraints to obtain sparse solutions.

Viewing PCA from a compression standpoint, the objective is to find the rank-K subspace projection $\mathbf{AA}^\top$ such that $\mathbf{A}^\top\mathbf{A} = \mathbf{I}$ ($\mathbf{A}$ is $p \times K$) which reconstructs a data point $\mathbf{x}$ as well as possible. This amounts to the following criterion,

$$\arg\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{XAA}^\top\|_F^2, \quad \text{such that} \quad \mathbf{A}^\top\mathbf{A} = \mathbf{I}.$$

The solution is readily available via a singular value decomposition; let $\mathbf{X} = \mathbf{UDV}^\top$, and set $\mathbf{A} = \mathbf{V}$.

Zou *et al.* (2006) show that this criterion can be relaxed into the following l_2-penalized formulation,

$$\arg\min_{\mathbf{A},\mathbf{B}} \|\mathbf{X} - \mathbf{X}\mathbf{A}\mathbf{B}^\top\|_F^2 + \delta\|\mathbf{B}\|_F^2, \quad \text{such that} \quad \mathbf{A}^\top\mathbf{A} = \mathbf{I},$$

where $\mathbf{B}$ is $p \times K$ and $\|\mathbf{B}\|_F^2 = \sum_{k=1}^K \|\beta_k\|_2^2$. After normalization such that each column of $\mathbf{B}$ has unit length, the optimal solution is $\mathbf{A} = \mathbf{B} = \mathbf{V}$, the loading matrix of PCA, irrespective of the choice of δ. The role of the ridge penalty on $\mathbf{B}$ is to provide unique solutions also when $p > n$, in which case δ must be non-zero. Since the loading vectors in $\mathbf{B}$ are orthogonal, we can estimate them sequentially by

$$\arg\min_{\alpha_k,\beta_k} \|\mathbf{X} - \mathbf{X}\beta_k\alpha_k^\top\|_F^2 + \delta\|\beta_k\|_2^2, \quad \text{subject to} \quad \mathbf{A}_k^\top\mathbf{A}_k = \mathbf{I}, \tag{23}$$

where $\mathbf{A}_k$ denotes the matrix $[\alpha_1 \dots \alpha_k]$. Aiming at an algorithm for computing a sparse matrix of loadings, we will now state an alternating algorithm for optimizing the above criterion for α_k and β_k. For this purpose, we have the following result.

Lemma 1 *Assume $\alpha_k^\top\alpha_k = 1$ and fix α_k, $\mathbf{X}$ and $\mathbf{Y}$. Then, the problems*

$$\arg\min_{\beta_k}\|\mathbf{Y} - \mathbf{X}\beta_k\alpha_k^\top\|_F^2, \qquad \arg\min_{\beta_k}\|\mathbf{Y}\alpha_k - \mathbf{X}\beta_k\|_2^2$$

have the same minimizer $\hat{\beta}_k = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}\alpha_k$.

This result (with $\mathbf{Y} = \mathbf{X}$ and adding the l_2 penalty) shows that for fixed α_k, the optimal β_k is given by $\hat{\beta}_k = (\mathbf{X}^\top\mathbf{X} + \delta\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{X}\alpha_k$. If we instead fix β_k, the optimal α_k is given by the following result.

Lemma 2 *Let $\mathbf{A}_{(k-1)}$ with $\mathbf{A}_{(k-1)}^\top\mathbf{A}_{(k-1)} = \mathbf{I}$ be the $(p \times k-1)$ matrix containing the first $k-1$ columns of $\mathbf{A}$. The "fix β_k, solve for α_k"-problem can then be formulated as,*

$$\hat{\alpha}_k = \arg\min_{\alpha_k} \|\mathbf{X} - \mathbf{X}\beta_k\alpha_k^\top\|_F^2 \quad \textit{subject to} \quad \alpha_k^\top\alpha_k = 1,\ \alpha_k^\top\mathbf{A}_{(k-1)} = \mathbf{0}. \tag{24}$$

Let $\mathbf{s} = \left(\mathbf{I} - \mathbf{A}_{(k-1)}\mathbf{A}_{(k-1)}^\top\right)\mathbf{X}^\top\mathbf{X}\beta_k$. Then, $\hat{\alpha}_k = \mathbf{s}/\sqrt{\mathbf{s}^\top\mathbf{s}}$.

Appendices B.1 and B.2 contain proofs of the above results. If applied alternately until convergence for each principal component, we end up with the full PCA solution. This convergence is assured since penalization (23) is convex and each alternating step leads to a lower function value.

Turning to the problem of estimating sparse principal components (a sparse loading matrix), an l_1 penalty is added to the formulation in Equation 23.

$$\{\hat{\alpha}_k, \hat{\beta}_k\} = \arg\min_{\alpha_k,\beta_k} \|\mathbf{X} - \mathbf{X}\beta_k\alpha_k^\top\|_F^2 + \delta\|\beta_k\|_2^2 + \lambda\|\beta_k\|_1, \quad \text{subject to} \quad \mathbf{A}_k^\top\mathbf{A}_k = \mathbf{I}. \tag{25}$$

Using the alternating approach defined above to optimize this criterion, we see that $\hat{\alpha}_k$ is estimated as before, while $\hat{\beta}_k$ is turned from a ridge regression problem into an elastic net problem. As before the response vector is $\mathbf{X}\alpha_k$. We arrive at Algorithm 4 for computing sparse principal components. This algorithm also handles the case where the l_2 regularization parameter

Algorithm 4 SPCA (Zou *et al.* 2006).

Let $K < p$ be the number of sparse principal loading vectors to estimate.
Let $\mathbf{A}$ be the $(p \times K)$ matrix consisting of the K first ordinary principal loading vectors.
for $k = 1, \ldots, K$ **do**
  while sparse loading vector β_k has not converged **do**
    if $\delta = \infty$ **then**
      Soft-thresholding: $\beta_k = \left(|\mathbf{X}^\top \mathbf{X} \alpha_k| - \lambda\right)_+ \operatorname{sign}(\mathbf{X}^\top \mathbf{X} \alpha_k)$.
    else
      Solve the elastic net problem $\beta_k = \arg\min_\beta \|\mathbf{X}\alpha_k - \mathbf{X}\beta\|^2 + \delta\|\beta\|^2 + \lambda\|\beta\|_1$.
    end if
    Normalize to unit length: $\beta_k = \beta_k / \sqrt{\beta_k^\top \beta_k}$.
    Update kth column of projection matrix $\mathbf{A}$: $\alpha_k = (\mathbf{I} - \mathbf{A}_{(k-1)}\mathbf{A}_{(k-1)}^\top)\mathbf{X}^\top \mathbf{X}\beta_k$.
    Normalize to unit length: $\alpha_k = \alpha_k / \sqrt{\alpha_k^\top \alpha_k}$.
  end while
end for
Output the coefficients $\mathbf{B} = [\beta_1 \ldots \beta_K]$.

δ is set to infinity. The elastic net estimation of β_k then turns into a soft-thresholding rule as described in Zou *et al.* (2006). This leads to a computational advantage, which is why this option is popular for very high-dimensional data arising from, e.g., image or gene expression data. It is our experience that this option also provides better solutions (in terms of explained variance for a fixed level of sparsity) in such cases.

Note that this algorithm is no longer convex, and may converge to local minima.

Since this is the first account of this sequential SPCA algorithm we give preliminary results of its performance and discuss advantages in relation to the previously proposed simultaneous approach (Zou *et al.* 2006).

A clear advantage of sequential estimation of components comes from running the algorithm once to estimate k components, and once to estimate $k + l$ components. The sequential approach will yield the exact same first k components in both cases whereas the simultaneous algorithm gives different results for all components.

Both algorithms are initialized with a matrix $\mathbf{A}$ equal to the loading matrix of regular PCA. The corresponding scores are ordered from high to low variance. The simultaneous approach often strays far from this initial solution and yields an arbitrary ordering in terms of variance of its components. In Sjöstrand, Stegmann, and Larsen (2006) we discuss several ways of establishing a sensible ordering of oblique components. The sequential algorithm is more likely to produce components of decreasing variance, and we have therefore chosen to return the components as-is, in order of computation.

The sequential approach transforms one large non-convex optimization problem into several small ones. Convergence rates for each such problem are typically orders of magnitude higher than that of the simultaneous approach. To verify this, we conducted an experiment on a synthetic data set of 600 observations, created from 200 observations each of three sparse components with added Gaussian noise. The total number of variables in the data set ranged from 10 to 1500 with increments of 10, and we ran the sequential and simultaneous algorithms once for each choice of p. We extracted three sparse principal components and compared com-

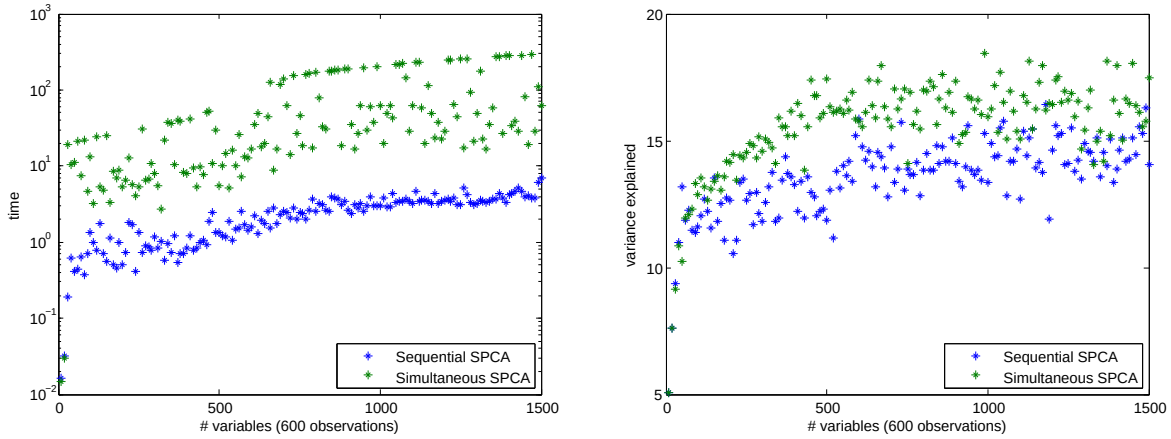

Figure 1: The figure on the left shows computation time for a data set with 600 observations and increasing dimensionality; 25 non-zero loadings were extracted. The sequential SPCA algorithm is faster by a factor 15–100. The figure on the right shows total adjusted variance for three components for the same data set. The sequential algorithm pays a small penalty for the one component at a time approach.

putation times and total adjusted variance. Figure 1 shows the results. The simultaneous algorithm was forced to give up after 1000 iterations, which occurred in a large proportion of runs. This is visible in the figure as a marked line of maximal computation times. Computation times for the sequential algorithm were lower by a factor 15–100 and with no premature terminations. The sequential algorithm is more restrictive than its sequential counterpart since previous components are fixed when estimating the next component. In our experience, one pays a small price in terms of variance for this restriction; this is shown in the right plot in Figure 1. We have not yet encountered a case were this reduction is significant.

4.5. Sparse linear discriminant analysis

Linear discriminant analysis (LDA) estimates orthogonal directions β_k in which observations $\mathbf{x}_i$ belonging to one of K classes are most separated. Separation is measured as the between-class variance σ_b^2 in relation to the within-class variance σ_w^2 of the projected data $\mathbf{X}\beta_k$. Class-belongings are dummy-encoded in a $(n \times K)$ matrix $\mathbf{Y}$ where element (i, j) is 1 if the ith observation belongs to the jth class, else 0. Further, the matrix $\mathbf{D}_\pi = \frac{1}{n}\mathbf{Y}^\top\mathbf{Y}$ is a diagonal matrix of class prior probabilities based on their frequency in $\mathbf{Y}$. Given these definitions, the matrix of class centroids is given by $\mathbf{M} = \frac{1}{n}\mathbf{D}_\pi^{-1}\mathbf{Y}^\top\mathbf{X}$, the total covariance matrix is $\Sigma = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$, the between-class covariance matrix is $\Sigma_b = \mathbf{M}^\top\mathbf{D}_\pi\mathbf{M} = \frac{1}{n}\mathbf{X}^\top\mathbf{Y}(\mathbf{Y}^\top\mathbf{Y})^{-1}\mathbf{Y}^\top\mathbf{X}$, and the within-class covariance matrix is $\Sigma_w = \Sigma - \Sigma_b = \frac{1}{n}\mathbf{X}^\top(\mathbf{I} - \mathbf{Y}(\mathbf{Y}^\top\mathbf{Y})^{-1}\mathbf{Y}^\top)\mathbf{X}$. The cost function to optimize for the kth direction is,

$$\arg\max_{\beta_k} \beta_k^\top \Sigma_b \beta_k \quad \text{subject to} \quad \beta_k^\top \Sigma_w \beta_k = 1, \beta_k^\top \Sigma_w \beta_l = 0, \forall l < k. \tag{26}$$

Standard differentiation leads to a an eigenvalue problem with respect to the matrix $\Sigma_w^{-1}\Sigma_b$, which yields the full set of solutions $\mathbf{B} = \{\beta_k\}$. Classification of a new observation $\mathbf{x}$ is performed by finding the closest centroid in the derived space defined by $\mathbf{B}$.

LDA relies on the assumptions that (1) the data is normally distributed and (2) all classes

have equal covariances. Although these assumptions are seldom met exactly, LDA often has as good or better performance compared to more flexible alternatives. This is in part due to the robustness of a method with few parameters to estimate. As with ordinary least squares in regression, there are situations where the inverse of the Gram matrix $\mathbf{X}^\top\mathbf{X}$ – which turns up in the estimation of regression coefficients as well as the estimation of Σ_w^{-1} – has high variance or is computationally infeasible. Similarly to ridge regression (cf., Section 4.3) the covariance matrix (or equivalently, the Gram matrix) may be replaced by a regularized variant $\Sigma + \delta\Omega$. If Ω is a positive definite matrix, then there exists a large-enough positive value of δ such that $\Sigma+\delta\Omega$ is positive definite (Petersen and Pedersen 2008). Application of this approach to LDA leads to *penalized* (linear) discriminant analysis (PDA; Hastie, Buja, and Tibshirani 1995); the estimate of Σ_w is simply replaced by $\Sigma_w + \delta\Omega$, otherwise the calculation and application proceeds as before. In the remainder of this section, we will use $\Omega = \mathbf{I}$ which shrinks the solutions towards those obtained by assuming a spherical common covariance matrix.

An alternative route to the solutions $\mathbf{B}$ of LDA/PDA is via *optimal scoring* (Hastie, Tibshirani, and Buja 1994). The PDA optimal scoring criterion is

$$\arg\min_{\Theta,\mathbf{B}} \|\mathbf{Y}\Theta - \mathbf{X}\mathbf{B}\|_F^2 + \delta\|\mathbf{B}\|_F^2 \quad \text{subject to} \quad \Theta^\top\mathbf{D}_\pi\Theta = \mathbf{I}. \tag{27}$$

The matrix Θ is a scoring matrix, orthogonal in $\mathbf{D}_\pi$, which assigns a multiple to each column (class) in $\mathbf{Y}$. This transformation of the dummy encodings circumvents problematic situations which otherwise make a regression approach to classification difficult (Hastie *et al.* 2009). As shown in detail in Hastie *et al.* (1995) and more succinctly in Hastie *et al.* (1994), the optimal solutions $\mathbf{B}$ are equivalent, up to a diagonal scaling matrix, to those obtained by PDA using the penalized within-class covariance matrix $\Sigma_w + \frac{\delta}{n}\mathbf{I}$. Differentiation, first with respect to $\mathbf{B}$ and then with respect to Θ gives the standard solution; first compute a multivariate ridge regression of $\mathbf{X}$ on $\mathbf{Y}$ yielding regression coefficients $\tilde{\mathbf{B}} = (\mathbf{X}^\top\mathbf{X} + \delta\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y}$. Θ is then obtained by an eigenanalysis of the matrix $\hat{\mathbf{Y}}\mathbf{Y}$, where $\hat{\mathbf{Y}} = \mathbf{X}\tilde{\mathbf{B}}$. The final directions $\mathbf{B}$ are obtained in a last step by $\mathbf{B} = \tilde{\mathbf{B}}\Theta$.

Similarly to the treatment of the PCA penalization (23), we can estimate the directions β_k sequentially since they are orthogonal. The estimation of the kth direction involves solving

$$\arg\min_{\theta_k,\beta_k} \|\mathbf{Y}\theta_k - \mathbf{X}\beta_k\|_2^2 + \delta\|\beta_k\|_2^2 \quad \text{subject to} \quad \Theta_k^\top\mathbf{D}_\pi\Theta_k = \mathbf{I}, \tag{28}$$

where Θ_k contains the k first columns of Θ. We will now describe an alternating algorithm which replaces the eigenanalysis-based recipe in the previous paragraph. This algorithm then naturally extends to *sparse* discriminant analysis, where the ridge regression estimate is replaced by an elastic net estimate. In line with Section 4.4, we first state the following lemma.

Lemma 3 *Let $\Theta_{(k-1)}$ with $\Theta_{(k-1)}^\top\mathbf{D}_\pi\Theta_{(k-1)} = \mathbf{I}$ be the $(p \times k-1)$ matrix containing the first $k-1$ columns of Θ. The "fix β_k, solve for θ_k"-problem can then be formulated as,*

$$\hat{\theta}_k = \arg\min_{\theta_k} \|\mathbf{Y}\theta_k - \mathbf{X}\beta_k\|_2^2 \quad \textit{subject to} \quad \theta_k^\top\mathbf{D}_\pi\theta_k = 1,\ \theta_k^\top\mathbf{D}_\pi\Theta_{(k-1)} = \mathbf{0}. \tag{29}$$

Let $\mathbf{s} = \left(\mathbf{I} - \Theta_{(k-1)}\Theta_{(k-1)}^\top\mathbf{D}_\pi\right)\mathbf{D}_\pi^{-1}\mathbf{Y}^\top\mathbf{X}\beta_k$. *Then,* $\hat{\alpha}_k = \mathbf{s}/\sqrt{\mathbf{s}^\top\mathbf{D}_\pi\mathbf{s}}$.

Proof Appendix B.2 gives a proof for Lemma 2; the proof for this lemma is equivalent, except the trivial addition of $\mathbf{D}_\pi$. The solution to this lemma is also given – sans proof – in Clemmensen *et al.* (2011).

Algorithm 5 SLDA (Clemmensen *et al.* 2011).

1: Let Q be the number of classes and $K < Q$ the number of discriminative directions.
2: Initialize $\mathbf{Y}$ an $n \times Q$ matrix of indicator variables with $Y_{i \in \text{class}_j j} = 1$, $\mathbf{D}_\pi = \frac{1}{n}\mathbf{Y}^\top\mathbf{Y}$, and $\Theta = \mathbf{I}_{(Q \times K)}$.
3: **for** $k = 1, \ldots, K$ **do**
4: **while** sparse discriminative direction β_k has not converged **do**
5: Solve the elastic net problem $\beta_k = \arg\min_\beta \|\mathbf{Y}\theta_k - \mathbf{X}\beta\|^2 + \delta\|\beta\|^2 + \delta\|\beta\|_1$.
6: Update kth column of Θ: $\theta_k = (\mathbf{I} - \Theta_{(k-1)}\Theta_{(k-1)}^\top\mathbf{D}_\pi)\mathbf{D}_\pi^{-1}\mathbf{Y}^\top\mathbf{X}\beta_k$.
7: Normalize to unit length: $\theta_k = \theta_k / \sqrt{\theta_k^\top\mathbf{D}_\pi\theta_k}$.
8: **end while**
9: **end for**
10: Output the coefficients $\mathbf{B} = [\beta_1 \ldots \beta_K]$.

The "fix θ_k, solve for β_k" problem is solved by the ridge regression estimate $\beta_k = (\mathbf{X}^\top\mathbf{X} + \delta\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y}\theta_k$.

We initialize Θ to the size $(K \times K)$ identity matrix. The directions are then obtained sequentially by alternating the estimation of β_k and θ_k until convergence. Convergence to a global optimum for each direction is guaranteed by the convexity of the cost function and its constraints and since each alteration is guaranteed to lower the cost.

With this algorithm in place, it is now straight-forward to extend it to include an l_1 penalty which promotes directions β_k which are sparse. This means that the space $\mathbf{B}$ in which the classification is carried out, consists of a subset of the available variables. As with all sparse methods, the possible benefits are ease of interpretation and (non-linear) suppression of noise. The l_1, l_2-regularized criterion is,

$$\{\hat{\theta}_k, \hat{\beta}_k\} = \arg\min_{\theta_k, \beta_k} \|\mathbf{Y}\theta_k - \mathbf{X}\beta_k\|_2^2 + \delta\|\beta_k\|_2^2 + \delta_k\|\beta_k\|_1 \quad \text{subject to} \quad \Theta_k^\top\mathbf{D}_\pi\Theta_k = \mathbf{I}, \tag{30}$$

which turns the ridge regression estimation of β_k from penalization (28) into an elastic net estimate. The resulting algorithm for sparse discriminant analysis is stated in Algorithm 5.

The normalization in Step 7 helps to avoid multiplicative drift towards the trivial solution where $\theta_k = \mathbf{0}$ and $\beta_k = \mathbf{0}$. To avoid additive drift we also require $\sum_i(\mathbf{D}_\pi\theta_k)_i = 0$, i.e., that θ_k is zero-mean in $\mathbf{D}_\pi$. In previous treatments of optimal scoring (see, e.g., Clemmensen *et al.* 2011), the columns of Θ are explicitly forced to be orthogonal to a vector of ones. However, it turns out that Step 6 implicitly guarantees this since $\mathbf{D}_\pi\theta_k = \mathbf{Y}^\top\mathbf{X}\beta_k - \mathbf{D}_\pi\Theta_{(k-1)}\Theta_{(k-1)}^\top\mathbf{D}_\pi\mathbf{D}_\pi^{-1}\mathbf{Y}^\top\mathbf{X}\beta_k$. The first term is clearly mean-zero since $\mathbf{X}$ is centered. The second term is mean-zero if $\mathbf{D}_\pi\Theta_{(k-1)}\Theta_{(k-1)}^\top\mathbf{D}_\pi\mathbf{D}_\pi^{-1}$ is mean zero. $\mathbf{D}_\pi\Theta_{(k-1)}\Theta_{(k-1)}^\top\mathbf{D}_\pi$ is the outer product of two mean-zero matrices and results in a mean-zero matrix. Multiplying this with the diagonal matrix $\mathbf{D}_\pi^{-1}$ does not change this property.

To allow for a more flexible model of the density of each class one may model each class as a mixture of Gaussian distributions. Clemmensen *et al.* (2011) describe an extension of the described algorithm which implements this.

5. Using the SpaSM toolbox

This section gives a brief overview of usage of the **SpaSM** toolbox. The examples covered can all be found in the source code. Issuing the command `help nameOfFunction` yields a detailed overview of the input and output to the functions, where `nameOfFunction` is a function in the toolbox, e.g., `forwardselection`.

After the code has been downloaded[1] one can add the path to the **SpaSM** directory in MATLAB to access the functions and scripts. The examples contain random generated data sets, the code for generating the data sets is also included in the examples.

A streamlined version of the examples is contained in the file `demo.m`. This includes the code to generate the data sets with appropriate seeds and all figures.

Most of the examples should work without issues in Octave (Eaton, Bateman, Hauberg, and Wehbring 2017). One needs to first install and load the **statistics** package.

5.1. Forward selection usage example

The example is contained in the file `example_forwardselection.m`. First a simulated data set is generated and pre-processed. This data set is referred to as simulated data 1. The code for generating the data set and the appropriate seed for the random number generator are inside the file.

The data set is described as follows. A set of six correlated mean zero predictors are generated from a multivariate random distribution. The covariance matrix has values of 1 in the diagonal and 0.6 in the off-diagonal entries. The response is generated as a linear combination of the first three predictors and i.i.d. Gaussian noise with standard deviation 2. The predictors are then centered and scaled to unit Euclidean length and the response is centered. The forward selection algorithm is then run with the following command:

```
>> [beta info] = forwardselection(X, y, 0, true, true);

Step        Added        Active set size
1               2                1
2               3                2
3               1                3
4               6                4
5               4                5
6               5                6
```

The inputs are the predictors variables `X`, the response variable `y`, a scalar `STOP` (which triggers early stopping if non-zero), a Boolean variable `STOREPATH` (which stores the values of the coefficients in the model estimated in each iteration) and a Boolean variable `VERBOSE` (which if `false` suppresses output to the command line).

The output to the command line shows what happened in each iteration, i.e., which variable was added and the size of the active set. The `beta` in the output is a matrix containing the coefficients in the model. Column i contains the parameters found in iteration i.

[1] The code is available at http://www.imm.dtu.dk/projects/spasm/ or as supplementary material.

```
>> beta

beta =

         0         0         0    9.4249   10.4411   10.2215   10.4525
         0   26.5885   16.2983   12.4341   13.3253   12.5398   12.5018
         0         0   15.2383   12.0823   13.2119   13.0880   13.4359
         0         0         0         0         0    2.6190    2.9017
         0         0         0         0         0         0   -1.2487
         0         0         0         0   -3.5534   -4.6301   -4.3471
```

The `info` variable in the output is a struct containing values for information criteria (see Section 4.1), degrees of freedom, steps and relative size of coefficients compared to a low bias model.

```
>> info

info =

    steps: 6
       df: [0 1 2 3 4 5 6]
       Cp: [241.363 51.658 19.405 8.533 9.012 10.182 12.000]
      AIC: [1.2588e+03 559.269 440.327 400.236 402.003 406.317 413.022]
      BIC: [1.2588e+03 568.876 459.542 429.057 440.431 454.353 470.664]
        s: [0 0.592 0.703 0.756 0.903 0.960 1.000]
```

Now we can find the best fitted model according to the *AIC*:

```
>> [bestAIC bestIdx] = min(info.AIC);
>> best_s = info.s(bestIdx);
```

This is a model containing variables 1, 2 and 3. All information criteria select the right model. Figure 2 shows how the values of the coefficients in the models change over the iterations.

5.2. Least angle regression usage example

We continue to use simulated data set 1 as for the forward selection algorithm. The example is contained in `example_lar.m`. We call the LAR algorithm with the following command:

```
>> [beta info] = lar(X, y, 0, true, true);

Step        Added        Active set size
1               2                1
2               3                2
3               1                3
4               4                4
5               6                5
6               5                6
```

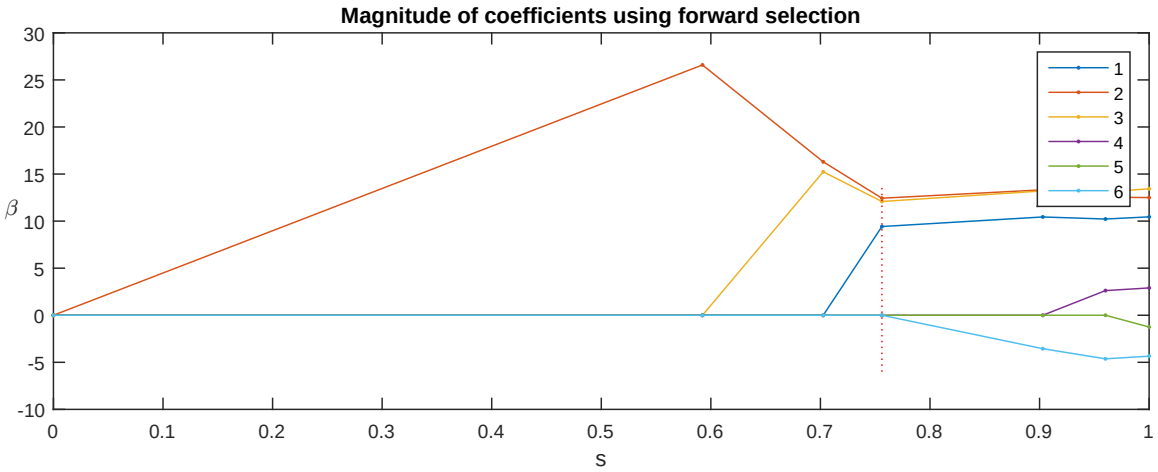

Figure 2: Value of parameters in models changing through the iterations of the forward selection algorithm on simulated data set 1. The x-axis shows the relative size of the parameter vector β compared to a low bias model. The dotted red vertical line indicates the model selected by AIC.

The input/output to the function and the output to the command line is virtually the same as for the forward selection algorithm. The `beta` matrix in the output is:

```
>> beta

beta =

         0         0         0    8.2840    8.8654    9.6638   10.4525
         0    1.0599    4.5161   11.4756   11.8199   12.2437   12.5018
         0         0    3.4562   11.0381   11.5741   12.4654   13.4359
         0         0         0         0    0.6168    1.7956    2.9017
         0         0         0         0         0         0   -1.2487
         0         0         0         0         0   -2.7259   -4.3471
```

Note that the model in the fourth column has the same parameters non-zero as in the forward selection algorithm. The values of the parameters are not the same and that is due to the fact that a new variable is added to the active set when it becomes equally important as an inactive variable.

The `info` struct holds the same information as for the forward selection algorithm. The output is:

```
>> info

info =

    steps: 6
       df: [0 1 2 3 4 5 6]
       Cp: [241.363 228.383 145.532 10.570 10.709 10.609 12.000]
      AIC: [1.2588e+03 1.2110e+03 905.449 407.747 408.262 407.894 413.022]
```

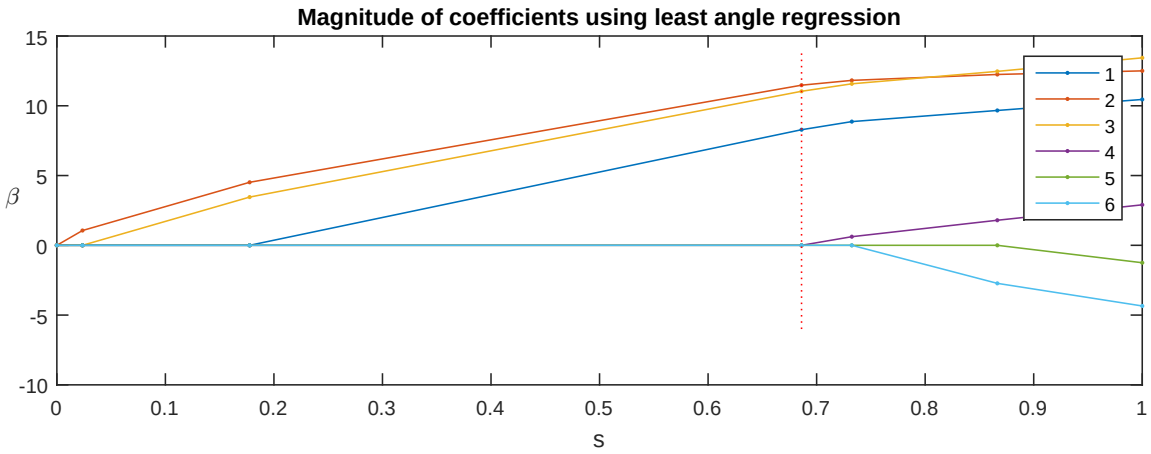

Figure 3: Value of parameters in models changing through the iterations of the LAR algorithm on simulated data set 1. The x-axis shows the relative size of the parameter vector β compared to a low bias model. The dotted red vertical line indicates the model selected by AIC.

```
BIC: [1.2588e+03 1.2206e+03 924.663 436.568 446.690 455.929 470.664]
  s: [0 0.024 0.178 0.686 0.732 0.866 1.000]
```

The first and last values of the information criteria are the same as for the forward selection but note that the intermediate values are quite different due to the different step sizes. Figure 3 shows a similar evolution of the parameters through the iterations as can be seen in Figure 2 for the forward selection algorithm. Note that the magnitude of the parameters of the selected model is relatively smaller than the one from forward selection.

5.3. The LASSO usage example

Again we use simulated data set 1 to try out the LASSO algorithm. The example is contained in the file `example_lasso.m`. The command and command line output are the following:

```
>> [beta info] = lasso(X, y, 0, true, true);

Step        Added        Dropped                Active set size
1               2                                       1
2               3                                       2
3               1                                       3
4               4                                       4
5               6                                       5
6               5                                       6
```

The only noticeable difference is the extra column named `Dropped`. In this example no variable is dropped from the active set, but that happens when a parameter value crosses zero. The input to the `lasso` function is the same as for `lar`. The values of `beta` and `info` are the same, since non of the active variables becomes zero throughout the iterations as can be seen in Figure 3.

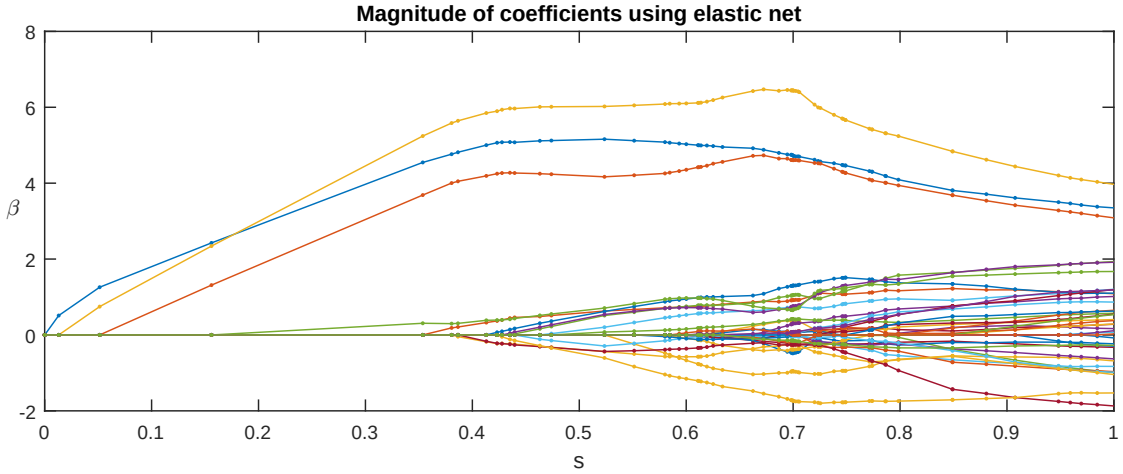

Figure 4: Value of parameters in models changing through the iterations of the elastic net algorithm on simulated data set 2. The x-axis shows the relative size of the parameter vector β compared to a low bias model.

5.4. The elastic net usage example

The elastic net can handle data sets with more variables than observations, given that the regularization parameter for the l_2 norm of the parameters is strictly positive. Here we create another simulated data set. We refer to this data set as simulated data set 2.

The data set is created as follows. We generate 30 observations of 40 variables. The predictors are sampled from a multivariate normal distribution with a similar covariance matrix as the simulated data set 1. The response is again a linear combination of the first three variables and i.i.d. Gaussian noise with standard deviation 0.5.

The example is contained in `example_elasticnet.m`. The algorithm is called as follows:

```
>> delta = 1e-3;
>> [beta info] = elasticnet(X, y, delta, 0, true, true);

Step        Added        Dropped              Active set size
1               1                                     1
2               3                                     2
3               2                                     3
4               26                                    4
5               9                                     5
6               31                                    6
...
```

Here `delta` is the regularization parameter for the l_2 norm. The additional output to the command line is omitted here. The values of the coefficients in the models through the iterations can be seen in Figure 4.

5.5. Sparse principal component analysis usage example

To demonstrate the usage of SPCA we generate another simulated data set with 1500 observations and 500 variables. This data set is referred to as simulated data set 3. First we

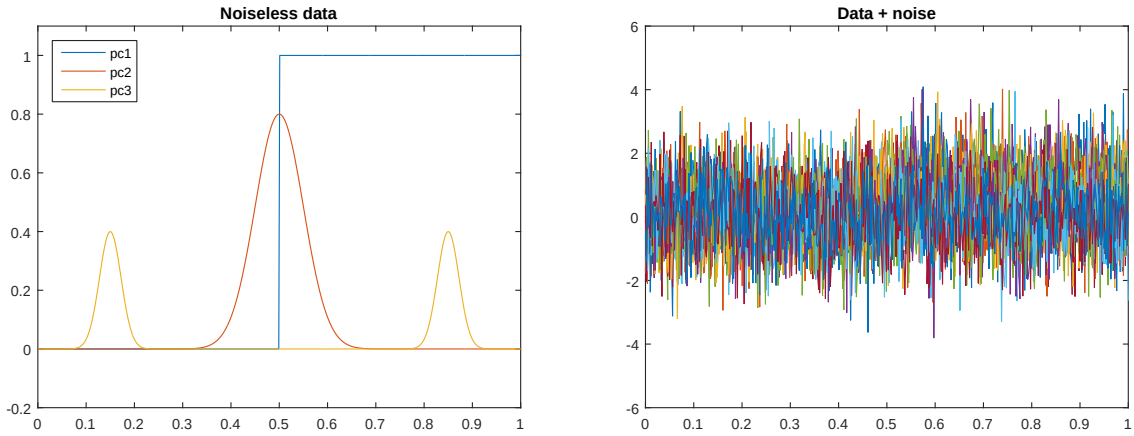

Figure 5: The figure on the left shows the three generated components, each consisting of 500 values. The figure on the right shows 5 noisy versions of each of the three components from simulated data set 3.

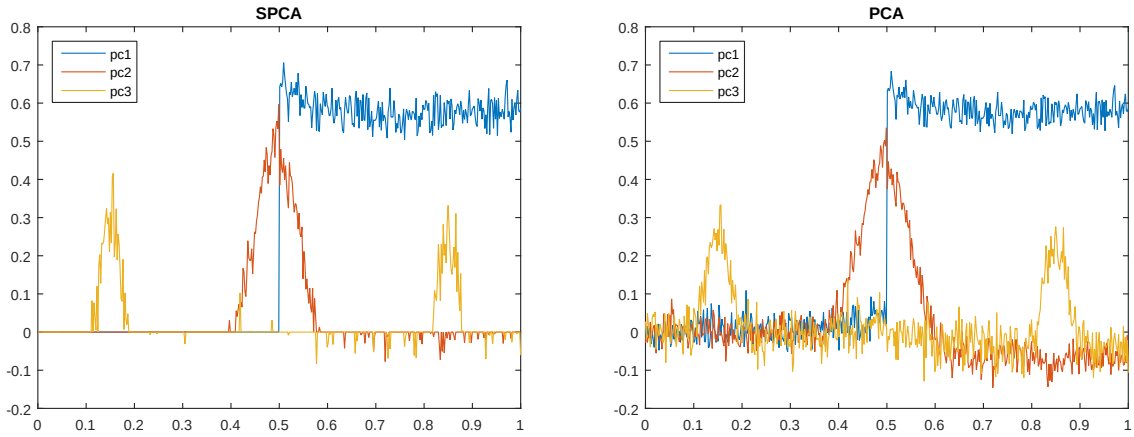

Figure 6: The figure on the left shows the results obtained by SPCA on simulated data set 3. The figure on the right shows the results from PCA.

generate 3 principal components and add i.i.d. Gaussian noise to them to generate 500 noisy versions of each. The three components can be seen without noise in Figure 5 on the left, and all the data with the noise is on the right. Note that each components has at least half the values equal to zero or close to zero. Code to generate simulated data set 3 with the appropriate seed can be found in the corresponding file `example_spca.m`. The data is stored in a matrix `X`.

To run the method we need to supply a few parameters. The parameters and call to the function in MATLAB are the following:

```
>> K = 3;
>> delta = inf;
>> stop = -[250 125 100];
```

```
>> maxiter = 3000;
>> convCriterion = 1e-9;
>> verbose = true;
>> [SL SD] = spca(X, [], K, delta, stop, maxiter, convCriterion, verbose);
```

The second argument can be the Gram matrix, otherwise the empty matrix is supplied. When the `stop` variable is negative, the absolute value is the number of desired non-zero values in each component. This is an advantage to other algorithms, where sparsity cannot directly be specified. The results compared to traditional PCA can be seen in Figure 6.

5.6. Sparse linear discriminant analysis usage example

To demonstrate the usage of SLDA we create yet another simulated data set, referred to as simulated data set 4. The training part of the data set consists of three classes with 100 observations from each class with 150 variables, the test part of the data set is identically sampled and of the same size. The data is generated from a multivariate normal distribution. The mean for the first class has the first ten variables as 0.6 and the rest as zero. The mean for the second class has variables 11–20 as 0.6 and the rest as zero and the third mean has variables 21–30 as 0.6 and the rest as zero. The classes have the same covariance matrix where there are 1s in the diagonal and a value of 0.6 in the off diagonal entries. The example is contained in the file `example_slda.m`. We can now run the algorithm with the following command:

```
>> [B theta] = slda(X, Y, delta, stop, Q, maxiter, tol, true);

Estimating direction 1
  Iteration: 10, convergence criterion: 0.025816
  Iteration: 20, convergence criterion: 2.7872e-06
  Iteration: 22, convergence criterion: 5.1135e-07
Estimating direction 2
  Iteration: 3, convergence criterion: 6.019e-30
```

The parameters for the function call are similar to the ones for SPCA. The parameter `Q` is the number of desired discriminative directions, in this case 2. The output consists of `B`, the regression parameters, and `theta`, the optimal scores.

We can now project the data onto the columns spanned by `B` and perform LDA. The training error is is 3.0% and test error is 5.3%. When we compare this to doing LDA on the raw data we get a training error of 1.0% and test error of 12.0%.

6. Example studies

The following examples elaborate on the differences of some of the algorithms presented in the **SpaSM** toolbox on real data sets. The first example demonstrates the difference of the LAR and LASSO algorithms, where the path of one of the coefficients crosses zero through the iterations. The second example demonstrates the difference of using PCA and SPCA on a shape data set consisting of male and female silhouettes. SPCA provides more local deviations from the mean which are easier to interpret.

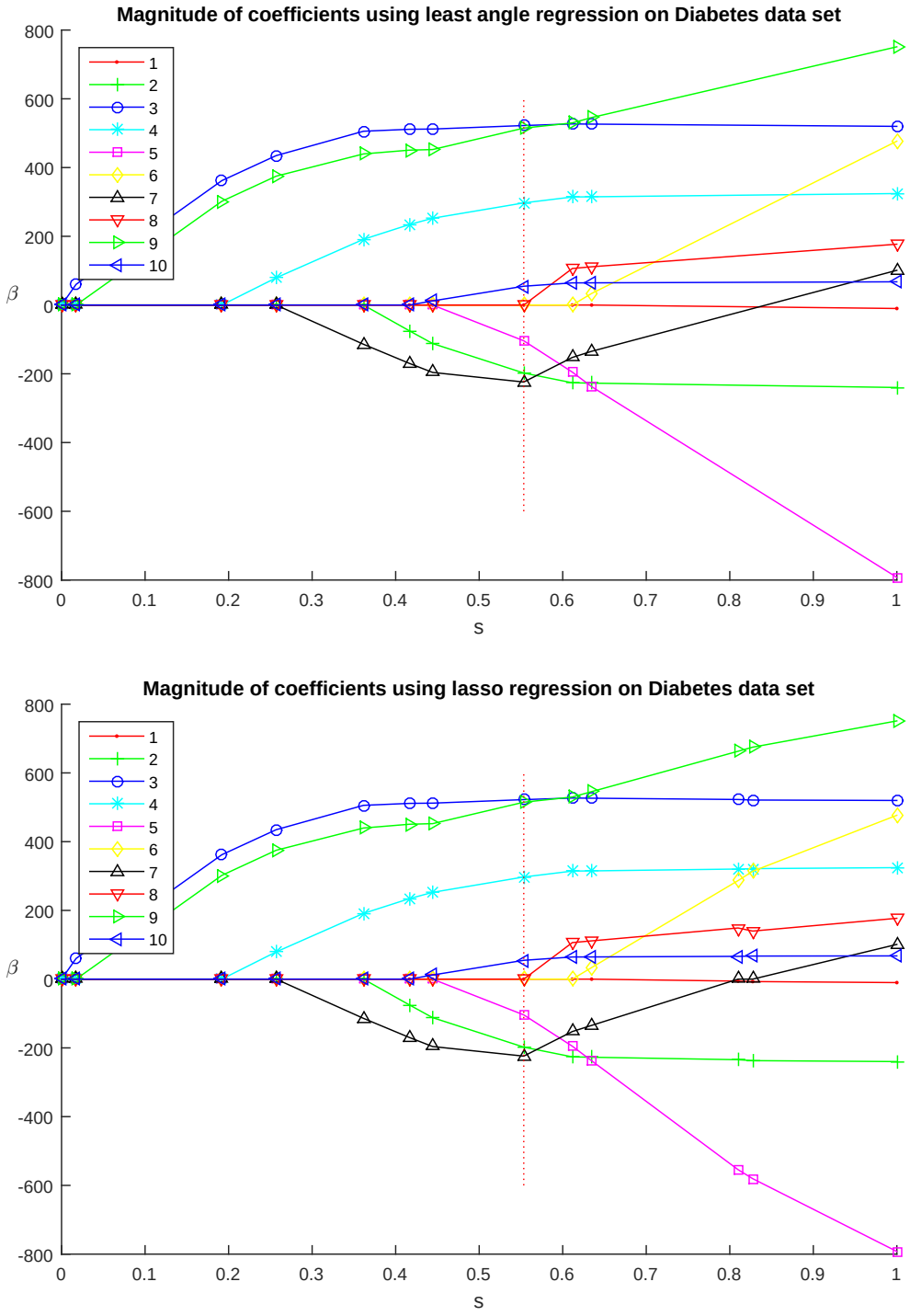

Figure 7: Coefficient paths for LAR and LASSO on the Diabetes data set. The red dotted line shows the model selected by AIC, which in this case is the same model for both. Note that variable 7 is the only one that changes sign. In the LASSO algorithm it is removed from the active set when this happens and then it re-enters in the next iteration.

6.1. Regression on the Diabetes data

To demonstrate the difference of the LASSO and the LAR algorithm we decided to use the Diabetes data set[2], see Efron *et al.* (2004). This data set is well known in the literature and it has one variable in it which changes sign through the iterations of LAR. This is what we need to demonstrate the difference between the two methods. The example is contained in the file `example_LarLasso.m`.

The data set includes 10 predictors, 1 response and 442 observations. The data set contains measurements on diabetic patients and the response is a quantitative measure of disease progression 1 year after the baseline.

The coefficient paths for the two methods can be seen in Figure 7. The plots are identical except for the fact that in the LASSO algorithm a variable is removed from the active set when it becomes zero.

6.2. PCA and SPCA on Silhouette data

This shape data set consists of silhouettes of 20 male and 19 female adults and was first presented in Thodberg and Olafsdottir (2003). Each silhouette consists of 65 points in 2D giving a total of 130 variables for each observation. The data has been aligned with Procrustes analysis prior to using the **SpaSM** toolbox. The example is contained in the file `example_spca_silDat.m`.

Running PCA shows that the first three components explain 82.9% of the variation in the data. Three components from SPCA with 40 non-zero variables each explain 65.27% of the variation in the data. The results can be seen in Figure 8.

The components obtained by PCA yield variation all over the silhouette. SPCA extracts the variables that explain most of the variation with the restriction that some of them need to be zero. These sparse components yield more local deformations on the silhouette, which are easier to interpret and to compare visually.

7. Collaboration and verification

We use tools and principles from software engineering in the development of this toolbox. A server-based repository (Apache Subversion, SVN; Apache Software Foundation 2011) allows toolbox authors to download (update in SVN terms) the latest toolbox snapshot, apply changes and then upload (commit) to the server when finished. Simultaneous editing of files is also possible where overlapping changes are merged in an intuitive way when committing changes back to the repository.

In software engineering *continuous integration* refers to the practice of committing small changes often to the repository, rather than scarce large updates. This keeps the effort required to merge changes from different authors to a minimum. To further improve the quality and effectiveness of the development, we employ *unit testing*. In parallel with the development of each toolbox entity, we develop several test scripts, each testing the one part (unit) of the code. As an example, one test file asserts (using the MATLAB `assert` command) that the elastic net with $\lambda = \delta = 0$ equals the ordinary least squares solution. Although we

[2]Available at https://web.stanford.edu/~hastie/Papers/LARS/diabetes.data.

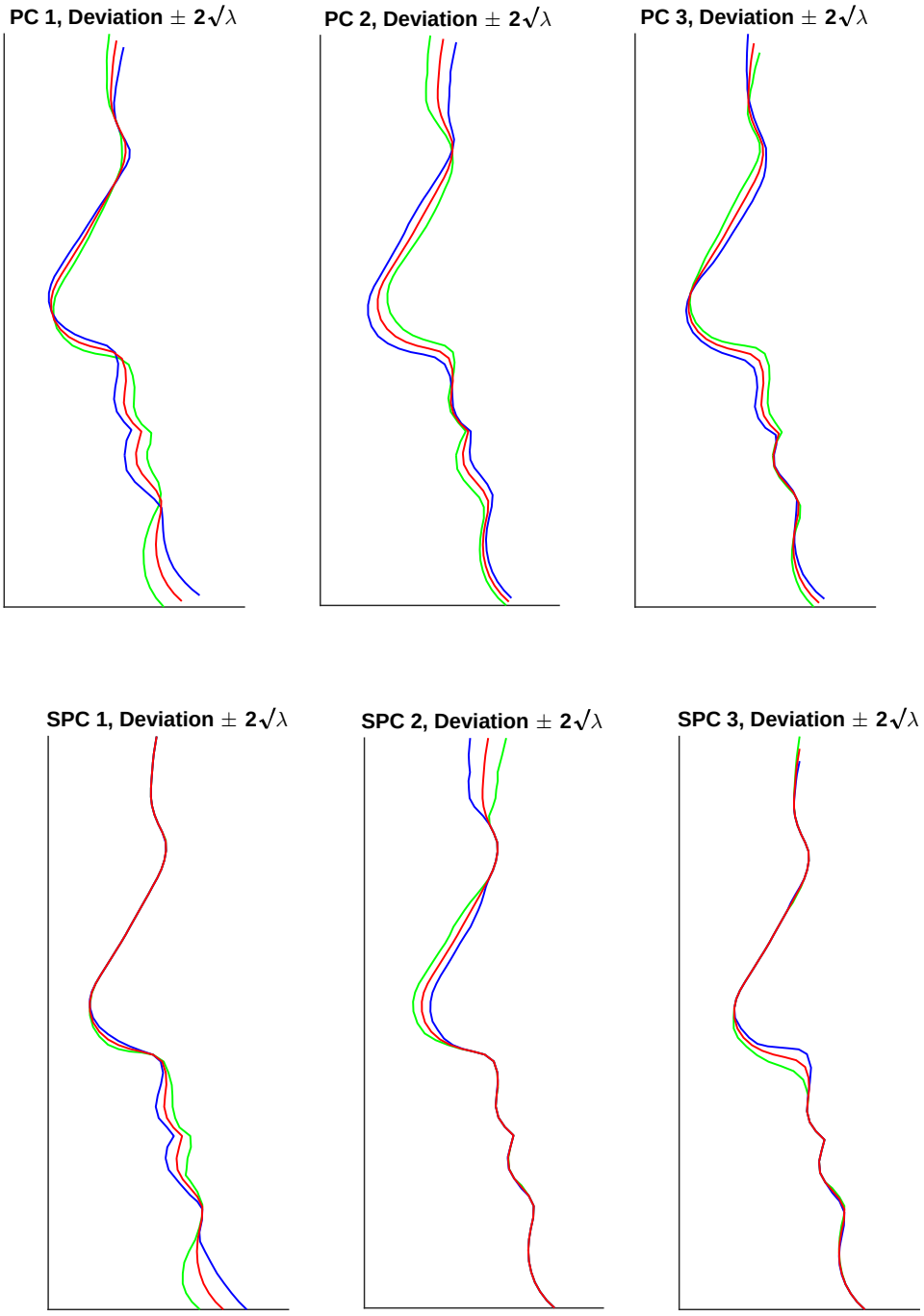

Figure 8: PCA and SPCA on the Silhouette data set. The top row shows deviations in the directions of the first three components from PCA and the second row shows them for the components obtained by SPCA. The red curve represents the mean shape, the green one represents -2 standard deviations from the mean and the blue $+2$ standard deviations from the mean.

Unit	Test	Acceptance criterion
LAR	Make sure the full LAR model is equal to the ordinary least squares model.	Results are equal.
LAR	Make sure LAR and LASSO are equal in cases where no variables are dropped in the LASSO.	Results are equal.
LAR	Profile code on $n \gg p$ and $p \gg n$ data sets.	Code has no apparent bottlenecks.
LASSO	Make sure the full LASSO model is equal to the ordinary least squares model.	Results are equal.
LASSO	Run with a data set with orthogonal predictor variables. Compare to soft-thresholding. Cf., Appendix A.	Results are equal.
LASSO	Profile code on $n \gg p$ and $p \gg n$ data sets.	Code has no apparent bottlenecks.
Elastic net	Make sure the full elastic net model is equal to the corresponding ridge regression model.	Results are equal.
Elastic net	Run with a data set with orthogonal predictor variables. Compare to soft-thresholding.	Results are equal.
Elastic net	Compare results with running LASSO with an elastic net-style augmented data matrix.	Results are equal.
Elastic net	Profile code on $n \gg p$ and $p \gg n$ data sets.	Code has no apparent bottlenecks.
SPCA	Compare the full SCPA model with that of a regular PCA. Try different values of δ.	Results are equal regardless of the value of δ.
SPCA	Profile code on $n \gg p$ and $p \gg n$ data sets.	Code has no apparent bottlenecks.
SLDA	Assert that the resulting optimal scores $\mathbf{Z} = \mathbf{Y}\theta$ are orthogonal.	$\mathbf{Z}^\top\mathbf{Z}/n = \mathbf{I}$.
SLDA	Compare the results of SLDA with no l_1 constraint to ridge regression on the matrix $\mathbf{Y}\theta$.	Results are equal.
SLDA	Compare the results of SLDA with no l_1 constraint to penalized discriminant analysis (LDA using the within-class covariance matrix $\Sigma_W + \frac{\delta}{n}\mathbf{I}$).	Results are equal.
SLDA	Profile code on $n \gg p$ and $p \gg n$ data sets.	Code has no apparent bottlenecks.
chol_insert	Compare updates of the Cholesky factorization to a direct Cholesky factorization of the corresponding matrices.	Results are equal.
chol_delete	Compare downdates of the Cholesky factorization to a direct Cholesky factorization of the corresponding matrices.	Results are equal.

Table 1: Toolbox unit tests.

currently have no automated procedure, we aim to run all such unit test files each time changes are uploaded to the repository. In this way, we get a strong indication to whether the new code is working as expected, and that uploaded changes did not break code that was working in an earlier version of the toolbox. Table 1 lists all relevant unit tests.

Other means of verification we have used are *code walkthrough*, a line-by-line inspection of finished code, and deployment of beta releases of the toolbox.

References

Alsahaf A (2015). "Sparse Principal Component Analysis Using Alternating Maximization." MATLAB Central File Exchange, URL http://www.mathworks.com/matlabcentral/fileexchange/47481-amjams-spca-am.

Apache Software Foundation (2011). "Apache Subversion." URL http://subversion.apache.org/.

Breheny P, Huang J (2011). "Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection." *The Annals of Applied Statistics*, **5**(1), 232. doi:10.1214/10-aoas388.

Cai T, Liu W, Luo X (2011). "A Constrained l_1 Minimization Approach to Sparse Precision Matrix Estimation." *Journal of the American Statistical Association*, **106**(494), 594–607. doi:10.1198/jasa.2011.tm10155.

Candes E, Tao T (2007). "The Dantzig Selector: Statistical Estimation When p is Much Larger than n." *The Annals of Statistics*, **35**(6), 2313–2351. doi:10.1214/009053606000001523.

Clemmensen L, Hastie T, Witten D, Ersbøll B (2011). "Sparse Discriminant Analysis." *Technometrics*, **53**(4), 406–413. doi:10.1198/tech.2011.08118.

Clemmensen L, Kuhn M (2016). ***sparseLDA**: Sparse Discriminant Analysis*. R package version 0.1-8, URL http://CRAN.R-project.org/package=sparseLDA.

Eaton JW, Bateman D, Hauberg S, Wehbring R (2017). *GNU Octave Version 4.2.1 Manual: A High-Level Interactive Language for Numerical Computations*. CreateSpace Independent Publishing Platform. URL https://www.gnu.org/software/octave/.

Efron B, Hastie T, Johnstone I, Tibshirani R (2004). "Least Angle Regression." *The Annals of Statistics*, **32**(2), 407–451. doi:10.1214/009053604000000067.

Friedman J, Hastie T, Tibshirani R (2008). "Sparse Inverse Covariance Estimation with the Graphical Lasso." *Biostatistics*, **9**(3), 432–441. doi:10.1093/biostatistics/kxm045.

Friedman J, Hastie T, Tibshirani R (2014). ***glasso**: Graphical Lasso- Estimation of Gaussian Graphical Models*. R package version 1.8, URL https://CRAN.R-project.org/package=glasso.

Friedman JH, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Fu WJ (1998). "Penalized Regressions: The Bridge versus the Lasso." *Journal of Computational and Graphical Statistics*, **7**(3), 397–416. doi:10.2307/1390712.

Goeman JJ (2010). "L_1 Penalized Estimation in the Cox Proportional Hazards Model." *Biometrical Journal*, **52**(1), 70–84. doi:10.1002/bimj.200900028.

Hastie T, Buja A, Tibshirani R (1995). "Penalized Discriminant Analysis." *The Annals of Statistics*, **23**(1), 73–102. doi:10.1214/aos/1176324456.

Hastie T, Efron B (2013). ***lars**: Least Angle Regression, Lasso and Forward Stagewise*. R package version 1.2, URL http://CRAN.R-project.org/package=lars.

Hastie T, Rosset S, Tibshirani R, Zhu J (2004). "The Entire Regularization Path for the Support Vector Machine." *Journal of Machine Learning Research*, **5**, 1391–1415.

Hastie T, Tibshirani R, Buja A (1994). "Flexible Discriminant Analysis by Optimal Scoring." *Journal of the American Statistical Association*, **89**(428), 1255–1270. doi:10.1080/01621459.1994.10476866.

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition. Springer-Verlag.

He S (2011). ***spaceExt**: Extension of SPACE*. R package version 1.0, URL https://CRAN.R-project.org/package=spaceExt.

Henson R (2013). *MATLAB R-Link*. MATLAB Central File Exchange, URL https://se.mathworks.com/matlabcentral/fileexchange/5051-matlab-r-link.

Hoerl AE, Kennard RW (1970). "Ridge Regression: Biased Estimation from Nonorthogonal Problems." *Technometrics*, **12**(1), 55–67. doi:10.1080/00401706.1970.10488634.

Kim SS (2009). "LARS Algorithm." MATLAB Central File Exchange, URL http://www.mathworks.com/matlabcentral/fileexchange/23186-lars-algorithm.

Kraemer N, Schaefer J, Boulesteix AL (2009). "Regularized Estimation of Large-Scale Gene Regulatory Networks Using Gaussian Graphical Models." *BMC Bioinformatics*, **10**(384), 1–24. doi:10.1186/1471-2105-10-384.

Le Y, Hastie T (2014). "Sparse Quadratic Discriminant Analysis and Community Bayes." arXiv:1407.4543 [stat.ML], URL https://arxiv.org/abs/1407.4543.

Li X, Zhao T, Wang L, Yuan X, Liu H (2014). ***flare**: Family of Lasso Regression*. R package version 1.5.0, URL https://CRAN.R-project.org/package=flare.

Liu J, Ji S, Ye J, *et al.* (2009). ***SLEP**: Sparse Learning with Efficient Projections*. Arizona State University. URL http://www.yelab.net/software/SLEP/.

Mai Q, Yang Y, Zou H (2015). ***msda**: Multi-Class Sparse Discriminant Analysis*. R package version 1.0.2, URL http://CRAN.R-project.org/package=msda.

Mai Q, Yang Y, Zou H (in press). "Multiclass Sparse Discriminant Analysis." *Statistica Sinica*. doi:10.5705/ss.202016.0117.

Meinshausen N, Bühlmann P (2006). "High-Dimensional Graphs and Variable Selection with the Lasso." *The Annals of Statistics*, **34**(3), 1436–1462. doi:10.1214/009053606000000281.

Osborne MR, Presnell B, Turlach BA (2000). "A New Approach to Variable Selection in Least Squares Problems." *IMA Journal of Numerical Analysis*, **20**(3), 389–403. doi:10.1093/imanum/20.3.389.

Pang H, Qi D, Liu H, Vanderbei R (2016). ***fastclime**: A Fast Solver for Parameterized LP Problems, Constrained L1 Minimization Approach to Sparse Precision Matrix Estimation and Dantzig Selector*. R package version 1.4.1, URL https://CRAN.R-project.org/package=fastclime.

Park MY, Hastie T (2007). "L_1-Regularization Path Algorithm for Generalized Linear Models." *Journal of the Royal Statistical Society B*, **69**(4), 659–677. doi:10.1111/j.1467-9868.2007.00607.x.

Petersen KB, Pedersen MS (2008). "The Matrix Cookbook." *Technical report*, Technical University of Denmark.

Qian J, Hastie T, Friedman J, Tibshirani R, Simon N (2014). "Glmnet for MATLAB." URL http://www.stanford.edu/~hastie/glmnet_matlab/.

Richtárik P, Takáč M, Ahipaşaoğlu SD (2012). "Alternating Maximization: Unifying Framework for 8 Sparse PCA Formulations and Efficient Parallel Codes." arXiv:1212.4137 [stat.ML], URL https://arxiv.org/abs/1212.4137.

Rosset S, Zhu J (2007). "Piecewise Linear Regularized Solution Paths." *The Annals of Statistics*, **35**(3), 1012–1030. doi:10.1214/009053606000001370.

Sjöstrand K, Stegmann MB, Larsen R (2006). "Sparse Principal Component Analysis in Medical Shape Modeling." In *Medical Imaging 2006: Image Processing*, volume 6144. doi:10.1117/12.651658.

Sun T (2013). ***scalreg**: Scaled Sparse Linear Regression*. R package version 1.0, URL https://CRAN.R-project.org/package=scalreg.

Sun T, Zhang CH (2012). "Scaled Sparse Linear Regression." *Biometrika*, **99**(4), 879–898. doi:10.1093/biomet/ass043.

The MathWorks Inc (2017). "MATLAB – The Language of Technical Computing, Version R2017b." URL http://www.mathworks.com/products/matlab/.

Thodberg HH, Olafsdottir H (2003). "Adding Curvature to Minimum Description Length Shape Models." In *British Machine Vision Conference, BMVC*.

Tibshirani R (1996). "Regression Shrinkage and Selection via the LASSO." *Journal of the Royal Statistical Society B*, **58**(1), 267–288.

Witten D (2015). ***penalizedLDA****: Penalized Classification Using Fisher's Linear Discriminant.* R package version 1.1, URL http://CRAN.R-project.org/package=penalizedLDA.

Witten DM, Tibshirani R (2011). "Penalized Classification Using Fisher's Linear Discriminant." *Journal of the Royal Statistical Society B*, **73**(5), 753–772. doi:10.1111/j.1467-9868.2011.00783.x.

Zhang CH (2010). "Nearly Unbiased Variable Selection under Minimax Concave Penalty." *The Annals of Statistics*, **38**(2), 894–942. doi:10.1214/09-aos729.

Zou H (2005). *Some Perspectives of Sparse Statistical Modeling.* Ph.D. thesis, Stanford University.

Zou H (2006). "The Adaptive Lasso and its Oracle Properties." *Journal of the American Statistical Association*, **101**(476), 1418–1429. doi:10.1198/016214506000000735.

Zou H, Hastie T (2005). "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society B*, **67**(2), 301–320. doi:10.1111/j.1467-9868.2005.00503.x.

Zou H, Hastie T, Tibshirani R (2006). "Sparse Principal Component Analysis." *Journal of Computational and Graphical Statistics*, **15**(2), 265–286. doi:10.1198/106186006x113430.

Zou H, Hastie T, Tibshirani R (2007). "On the "Degrees of Freedom" of the Lasso." *The Annals of Statistics*, **35**(5), 2173–2192. doi:10.1214/009053607000000127.

A. Sparse regression with orthogonal predictors

Several methods have simple closed-form solutions in cases where the predictor variables are orthogonal and have Euclidean length 1. Often, the estimation can be split into p separate problems, one for each β_i. We will quickly review how this works for the LASSO, the treatment is similar for the elastic net and LAR. We use this property for testing purposes, cf., Table 1.

$$\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\|\beta\|_1 \Longleftrightarrow \|\mathbf{y} - \mathbf{x}_i\beta_i\|^2 + \lambda|\beta_i|, \forall i.$$

Optimizing the expression for a single $\hat{\beta}_i$ involves taking first derivatives and setting to zero,

$$-2\mathbf{x}_i^\top(\mathbf{y} - \mathbf{x}_i\beta_i) + \lambda \cdot \text{sign}(\beta_i) = 0, i \in \mathcal{A}.$$

Using $\mathbf{x}_i^\top\mathbf{y} = \beta_i^{\text{OLS}}$ and $\mathbf{x}_i^\top\mathbf{x}_i = 1$ we have,

$$-2\beta_i^{\text{OLS}} + 2\beta_i + \lambda \cdot \text{sign}(\beta_i) = 0, i \in \mathcal{A}.$$

For sufficiently large values of λ, β_i will shrink to exactly zero. For any other value of λ, β_i will agree in sign with β_i^{OLS}. Therefore, we have,

$$\beta_i = \text{sign}(\beta_i^{\text{OLS}})\left(|\beta_i^{\text{OLS}}| - \frac{\lambda}{2}\right)^+, \forall i,$$

where $(\cdot)^+$ denotes the hinge function $\max(\cdot, 0)$.

B. Proofs

B.1. Proof of Lemma 1

Using $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$, $\text{tr}(\mathbf{A} + \mathbf{B}) = \text{tr}(\mathbf{A}) + \text{tr}(\mathbf{B})$ and $\alpha_k^\top\alpha_k = 1$ we have,

$$\begin{aligned}
\|\mathbf{X} - \mathbf{X}\beta_k\alpha_k^\top\|_F^2 &= \text{tr}\left(\mathbf{X}^\top\mathbf{X} + \alpha_k\beta_k^\top\mathbf{X}^\top\mathbf{X}\beta_k\alpha_k^\top - 2\mathbf{X}^\top\mathbf{X}\beta_k\alpha_k^\top\right) \\
&= \text{tr}(\mathbf{X}^\top\mathbf{X}) + \text{tr}(\mathbf{X}\beta_k\alpha_k^\top\alpha_k\beta_k^\top\mathbf{X}^\top) - 2\alpha_k^\top\mathbf{X}^\top\mathbf{X}\beta_k \\
&= \text{tr}(\mathbf{X}^\top\mathbf{X}) + \text{tr}(\mathbf{X}\beta_k\beta_k^\top\mathbf{X}^\top) - 2\alpha_k^\top\mathbf{X}^\top\mathbf{X}\beta_k \\
&= \text{tr}(\mathbf{X}^\top\mathbf{X}) + \beta_k^\top\mathbf{X}^\top\mathbf{X}\beta_k - 2\alpha_k^\top\mathbf{X}^\top\mathbf{X}\beta_k,
\end{aligned}$$

which clearly has the same minimizing β_k as

$$\|\mathbf{X}\alpha_k - \mathbf{X}\beta_k\|_2^2 = \alpha_k^\top\mathbf{X}^\top\mathbf{X}\alpha_k + \beta_k^\top\mathbf{X}^\top\mathbf{X}\beta_k - 2\alpha_k^\top\mathbf{X}^\top\mathbf{X}\beta_k.$$

Differentiation of any of the expressions gives $\hat{\beta}_k = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top(\mathbf{X}\alpha_k)$. This proof is also detailed in a slightly different context by Zou *et al.* (2006).

B.2. Proof of Lemma 2

Incorporating the constraints into the cost function in Equation 24 using Lagrange multipliers λ (length $k-1$ vector) and γ (scalar) the problem becomes

$$\arg\min_{\alpha_k} \text{tr}\left[\alpha_k\beta_k^\top\mathbf{X}^\top\mathbf{X}\beta_k\alpha_k^\top - 2\alpha_k\beta_k^\top\mathbf{X}^\top\mathbf{X} + \mathbf{X}^\top\mathbf{X}\right] + \alpha_k^\top\mathbf{A}_{(k-1)}\lambda + \gamma(\alpha_k^\top\alpha_k - 1).$$

Differentiating and setting to zero, and solving for α_k leads to

$$\hat{\alpha}_k = \frac{1}{\beta_k^\top \mathbf{X}^\top \mathbf{X} \beta_k + \gamma} \left[\mathbf{X}^\top \mathbf{X} \beta_k - \frac{1}{2} \mathbf{A}_{(k-1)} \lambda \right],$$

or equivalently,

$$\hat{\alpha}_k = \frac{1}{\beta} \left[\mathbf{X}^\top \mathbf{X} \beta_k - \mathbf{A}_{(k-1)} \alpha \right]. \tag{31}$$

The orthogonality constraints give

$$\mathbf{A}_{(k-1)}^\top \hat{\alpha}_k = 0 \Leftrightarrow \frac{1}{\beta} \left[\mathbf{A}_{(k-1)}^\top \mathbf{X}^\top \mathbf{X} \beta_k - \alpha \right] = 0 \Leftrightarrow \alpha = \mathbf{A}_{(k-1)}^\top \mathbf{X}^\top \mathbf{X} \beta_k.$$

Inserting this expression for α into Equation 31 and simplifying gives

$$\hat{\alpha}_k = \frac{1}{\beta} \left(\mathbf{I} - \mathbf{A}_{(k-1)} \mathbf{A}_{(k-1)}^\top \right) \mathbf{X}^\top \mathbf{X} \beta_k \equiv \frac{1}{\beta} \mathbf{s}.$$

Finally, the constraint $\alpha_k^\top \alpha_k = 1$ gives $\beta = \sqrt{\mathbf{s}^\top \mathbf{s}}$ such that $\hat{\alpha}_k = \mathbf{s} / \sqrt{\mathbf{s}^\top \mathbf{s}}$. In practice, we first calculate $\mathbf{s}$ and then normalize this vector to unit length.

Affiliation:

Karl Sjöstrand
EXINI Diagnostics AB
Ideon Science Park
Gateway, Scheelevägen 27
SE-223 70 Lund, Sweden
E-mail: karl.sjostrand@exini.com
URL: http://www.imm.dtu.dk/projects/spasm/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
April 2018, Volume 84, Issue 10 *Submitted:* 2015-10-09
doi:10.18637/jss.v084.i10 *Accepted:* 2017-06-14