Mach Learn (2011) 85:3–39
DOI 10.1007/s10994-010-5226-3

A majorization-minimization approach to the sparse generalized eigenvalue problem

Bharath K. Sriperumbudur · David A. Torres · Gert R.G. Lanckriet

Received: 1 March 2010 / Revised: 17 August 2010 / Accepted: 3 November 2010 /
Published online: 7 December 2010
© The Author(s) 2010. This article is published with open access at Springerlink.com

Abstract Generalized eigenvalue (GEV) problems have applications in many areas of science and engineering. For example, principal component analysis (PCA), canonical correlation analysis (CCA) and Fisher discriminant analysis (FDA) are specific instances of GEV problems, that are widely used in statistical data analysis. The main contribution of this work is to formulate a general, efficient algorithm to obtain sparse solutions to a GEV problem. Specific instances of sparse GEV problems can then be solved by specific instances of this algorithm. We achieve this by solving the GEV problem while constraining the cardinality of the solution. Instead of relaxing the cardinality constraint using a ℓ_1-norm approximation, we consider a tighter approximation that is related to the negative log-likelihood of a Student's t-distribution. The problem is then framed as a d.c. (difference of convex functions) program and is solved as a sequence of convex programs by invoking the majorization-minimization method. The resulting algorithm is proved to exhibit *global convergence* behavior, i.e., for any random initialization, the sequence (subsequence) of iterates generated by the algorithm converges to a stationary point of the d.c. program. Finally, we illustrate the merits of this general sparse GEV algorithm with three specific examples of sparse GEV problems: sparse PCA, sparse CCA and sparse FDA. Empirical evidence for these examples suggests that the proposed sparse GEV algorithm, which offers a general framework

Editors: Süreyya Özöğür-Akyüz, Devrim Ünay, and Alex Smola.

B.K. Sriperumbudur (✉) · G.R.G. Lanckriet
Department of Electrical and Computer Engineering, University of California, La Jolla, San Diego, CA 92093-0407, USA
e-mail: bharathsv@ucsd.edu

G.R.G. Lanckriet
e-mail: gert@ece.ucsd.edu

D.A. Torres · G.R.G. Lanckriet
Department of Computer Science and Engineering, University of California, La Jolla, San Diego, CA 92093-0407, USA

D.A. Torres
e-mail: datorres@cs.ucsd.edu

to solve any sparse GEV problem, will give rise to competitive algorithms for a variety of applications where specific instances of GEV problems arise.

Keywords Generalized eigenvalue problem · Principal component analysis · Canonical correlation analysis · Fisher discriminant analysis · Sparsity · D.c. program · Majorization-minimization · Zangwill's theory of global convergence · Music annotation · Cross-language document retrieval

1 Introduction

The generalized eigenvalue (GEV) problem for the matrix pair $(\boldsymbol{A}, \boldsymbol{B})$ is the problem of finding a pair $(\lambda, \boldsymbol{x})$ such that

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}, \tag{1}$$

where $\boldsymbol{A}$, $\boldsymbol{B} \in \mathbb{R}^{n\times n}$, $\mathbb{R}^n \ni \boldsymbol{x} \neq \boldsymbol{0}$ and $\lambda \in \mathbb{R}$. When $\boldsymbol{B}$ is an identity matrix, the problem in (1) is simply referred to as an eigenvalue problem. Eigenvalue problems are so fundamental that they have applications in almost every area of science and engineering (Strang 1986).

In multivariate statistics, GEV problems are prominent and appear in problems dealing with high-dimensional data analysis, visualization and pattern recognition. In these applications, usually $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{S}^n$ (the set of symmetric matrices of size $n \times n$ defined over $\mathbb{R}$) and $\boldsymbol{B} \in \mathbb{S}^n_{++}$ (set of positive definite matrices of size $n \times n$ defined over $\mathbb{R}$). The variational formulation for the GEV problem in (1) is given by

$$\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B}) = \max_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} \,:\, \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} = 1 \right\}, \tag{GEV-P}$$

where $\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B})$ is the maximum generalized eigenvalue associated with the matrix pair, $(\boldsymbol{A}, \boldsymbol{B})$. The $\boldsymbol{x}$ that maximizes (GEV-P) is called the generalized eigenvector associated with $\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B})$. Some of the well-known and widely used data analysis techniques that are specific instances of (GEV-P) are:

(a) Principal component analysis (PCA) (Hotelling 1933; Jolliffe 1986), a classic tool for data analysis, data compression and visualization, finds the direction of maximal variance in a given multivariate data set. This technique is used in dimensionality reduction wherein the ambient space in which the data resides is approximated by a low-dimensional subspace without significant loss of information. The variational form of PCA is obtained by choosing $\boldsymbol{A}$ to be the covariance matrix (which is a positive semi-definite matrix defined over $\mathbb{R}$) associated with the multivariate data and $\boldsymbol{B}$ to be the identity matrix in (GEV-P).

(b) Canonical correlation analysis (CCA) (Hotelling 1936), similar to PCA, is also a data analysis and dimensionality reduction method. However, while PCA deals with only one data space $\mathcal{X}$ (from which the multivariate data is obtained), CCA proposes a way for dimensionality reduction by taking into account relations between samples from two (or more) spaces $\mathcal{X}$ and $\mathcal{Y}$. The assumption is that the data points from these two spaces contain some joint information that is reflected in correlations between them. Directions along which this correlation is high are thus assumed to be relevant directions when these relations are to be captured. The variational formulation for CCA is given by

$$\max_{\boldsymbol{w}_x \neq \boldsymbol{0},\, \boldsymbol{w}_y \neq \boldsymbol{0}} \frac{\boldsymbol{w}_x^T \boldsymbol{\Sigma}_{xy} \boldsymbol{w}_y}{\sqrt{\boldsymbol{w}_x^T \boldsymbol{\Sigma}_{xx} \boldsymbol{w}_x}\sqrt{\boldsymbol{w}_y^T \boldsymbol{\Sigma}_{yy} \boldsymbol{w}_y}}, \tag{2}$$

where $\boldsymbol{w}_x$ and $\boldsymbol{w}_y$ are the directions in $\mathcal{X}$ and $\mathcal{Y}$ along which the data is maximally correlated. $\boldsymbol{\Sigma}_{xx}$ and $\boldsymbol{\Sigma}_{yy}$ represent the covariance matrices for $\mathcal{X}$ and $\mathcal{Y}$ respectively and $\boldsymbol{\Sigma}_{xy} = \boldsymbol{\Sigma}_{yx}^T$ represents the cross-covariance matrix between $\mathcal{X}$ and $\mathcal{Y}$. Equation (2) can be rewritten as

$$\max_{\boldsymbol{w}_x, \boldsymbol{w}_y} \{\boldsymbol{w}_x^T \boldsymbol{\Sigma}_{xy} \boldsymbol{w}_y \,:\, \boldsymbol{w}_x^T \boldsymbol{\Sigma}_{xx} \boldsymbol{w}_x = 1,\ \boldsymbol{w}_y^T \boldsymbol{\Sigma}_{yy} \boldsymbol{w}_y = 1\}, \tag{3}$$

which in turn can be written in the form of (GEV-P) with:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{0} \end{pmatrix}, \qquad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_{yy} \end{pmatrix} \quad \text{and} \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{w}_x \\ \boldsymbol{w}_y \end{pmatrix}.$$

(c) In the binary classification setting, Fisher discriminant analysis (FDA) finds a one-dimensional subspace, $\boldsymbol{w} \in \mathbb{R}^n$, the projection of data onto which leads to maximal separation between the classes. Let $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ denote the mean vector and covariance matrix associated with class i. The variational formulation of FDA is given by

$$\max_{\boldsymbol{w} \neq \boldsymbol{0}} \frac{(\boldsymbol{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2))^2}{\boldsymbol{w}^T(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)\boldsymbol{w}},$$

which can be rewritten as

$$\begin{aligned} \max_{\boldsymbol{w}} \quad & \boldsymbol{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{w} \\ \text{s.t.} \quad & \boldsymbol{w}^T(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)\boldsymbol{w} = 1. \end{aligned} \tag{4}$$

Therefore, the FDA formulation is similar to (GEV-P) with $\boldsymbol{A} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$, called the *between-cluster variance* and $\boldsymbol{B} = \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2$, called the *within-cluster variance*. For multi-class problems, similar formulations lead to multiple-discriminant analysis.

Despite the simplicity and popularity of these data analysis and modeling methods, one potential limitation (depending on the application) is the lack of sparsity in their solution. Their solution vector, i.e., $\boldsymbol{x}$, is a linear combination of all input variables. Often, this makes it difficult to interpret the results and, therefore, limits their use for applications where interpretation of the solution is important. In the following, we motivate the need for sparse solutions to GEV problems by explaining how, for some specific instances of GEV problems, i.e., PCA, CCA and FDA, a variety of applications could benefit from sparsity for different reasons.

In many PCA applications, the coordinate axes have a physical interpretation; in biology, for example, each axis might correspond to a specific gene. In these cases, the interpretation of the principal components would be facilitated if they contained only few non-zero entries (or, loadings) while explaining most of the variance in the data. Moreover, in certain applications, e.g., financial asset trading strategies based on PCA techniques, the sparsity of the solution has important consequences, since fewer non-zero loadings imply fewer transaction costs. For CCA, consider a document translation application where two copies of a corpus of documents, one written in English and the other in German are given. The goal is to extract multiple low-dimensional representations of the documents, one in each language, each explaining most of the variation in the documents of a single language while maximizing the correlation between the representations to aid translation. Sparse representations, equivalent to representing the documents with a small set of words in each language, would allow to interpret the underlying translation mechanism and model it better. In music

annotation, CCA can be applied to model the correlation between semantic descriptions of songs (e.g., reviews) and their acoustic content. Sparsity in the semantic canonical components would allow to select the most meaningful words to describe musical content. This is expected to improve music annotation and retrieval systems. In a classification setting like FDA, feature selection aids generalization performance by promoting sparse solutions. To summarize, sparse representations are generally desirable as they aid human understanding, reduce computational and economic costs and promote better generalization.

In this paper, we propose an algorithm to find *sparse solutions to the generalized eigenvalue problem*. Specific instances of the sparse GEV problem, like, e.g., sparse CCA, can then be solved by specific instances of this algorithm. The sparse GEV problem can be written as

$$\max_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \,:\, \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} = 1,\ \|\boldsymbol{x}\|_0 \leq k \right\}, \tag{SGEV-P}$$

where $\boldsymbol{A} \in \mathbb{S}^n$, $\boldsymbol{B} \in \mathbb{S}^n_{++}$, $1 \leq k \leq n$ and $\|\boldsymbol{x}\|_0$ denotes the cardinality of $\boldsymbol{x}$, i.e., the number of non-zero elements of $\boldsymbol{x}$. Depending on the application and its sparsity requirements, k is chosen a priori. Equation (SGEV-P) can be solved either as a continuous optimization problem after relaxing the cardinality constraint or as a discrete optimization problem. In this paper, we follow the former approach, leading to our main contribution, a general algorithm for sparse GEV problems, which is presented in Sect. 3.

The first step in solving (SGEV-P) as a continuous optimization problem is to approximate the cardinality constraint. One usual heuristic is to approximate $\|\boldsymbol{x}\|_0$ by $\|\boldsymbol{x}\|_1$ (see Sect. 2 for the details on notation). Building on the earlier version of our work (Sriperumbudur et al. 2007), in Sect. 3.1, we approximate the cardinality constraint in (SGEV-P) as the negative log-likelihood of a Student's t-distribution, which is a tighter approximation than $\|\boldsymbol{x}\|_1$, and has been used earlier in many different contexts (Weston et al. 2003; Fazel et al. 2003; Candes et al. 2007). We then formulate this approximate problem as a d.c. (difference of convex functions) program and solve it in Sect. 3.3 as a sequence of quadratically constrained quadratic programs (QCQPs) (Boyd and Vandenberghe 2004), using the majorization-minimization (MM) method (Lange et al. 2000; Hunter and Lange 2004), which is briefly discussed in Sect. 3.2. As a special case, when $\boldsymbol{A} \in \mathbb{S}^n_+$ (the set of positive semidefinite matrices of size $n \times n$ defined over $\mathbb{R}$) and $\boldsymbol{B} \in \mathbb{S}^n_{++}$ is a diagonal matrix (as is the case, e.g., for PCA, where $\boldsymbol{B}$ is an identity matrix), a simple iterative update rule can be obtained in closed form, which has a per iteration complexity of $O(n^2)$. Since the algorithm presented in this paper holds for any $\boldsymbol{A} \in \mathbb{S}^n$, it is more general than the ones in Sriperumbudur et al. (2007) and Moghaddam et al. (2007a), where $\boldsymbol{A} \in \mathbb{S}^n_+$. For example, the algorithm in Sriperumbudur et al. (2007) cannot handle the sparse CCA problem as $\boldsymbol{A} \in \mathbb{S}^n$ is indefinite. A preliminary version of the algorithm proposed in this paper, appeared in Torres et al. (2007b) and Sriperumbudur et al. (2009).

Since the proposed algorithm is an iterative procedure, using results from the global convergence theory of iterative algorithms (Zangwill 1969), we show in Sect. 3.4 that it is *globally convergent*. I.e., for any random initialization, the sequence (subsequence) of iterates generated by the algorithm converges to a stationary point of the d.c. program (see Sect. 3.4 for a detailed definition). A complete convergence analysis, including the rate of convergence and the quality of the obtained solution, is beyond the scope of this work and is the subject of future work.

In Sect. 4, we illustrate the merits of the proposed sparse GEV algorithm for some specific choices of $\boldsymbol{A}$ and $\boldsymbol{B}$: algorithms for sparse PCA, sparse CCA and sparse FDA are obtained as special cases of the general sparse GEV algorithm. The empirical results observed for these special cases demonstrate the promise of the proposed algorithm, which is

more general than any of the special cases. More specifically, in Sect. 4.1, we compare the specific instance of our algorithm for $\boldsymbol{A} \in \mathbb{S}^n_+$ and $\boldsymbol{B}$ the identity matrix (which we call DC-PCA) to other sparse PCA algorithms, SPCA (Zou et al. 2006), DSPCA (d'Aspremont et al. 2007), GSPCA (Moghaddam et al. 2007b) and GPower$_{\ell_0}$ (Journée et al. 2010), in terms of sparsity vs. explained variance on the "pit props" benchmark dataset, a random test dataset and three high-dimensional datasets, where the goal is to find relevant genes (as few as possible) while explaining the maximum possible variance. The results show that DC-PCA is competitive (with respect to sparsity, explained variance as well computational efficiency) with the state-of-the-art for sparse PCA. Given that sparse PCA is a well-studied problem, this indicates the potential of the general sparse GEV framework, proposed in this work, for instances of sparse GEV problems that have not been studied extensively. In Sect. 4.2, the proposed sparse GEV algorithm is used in two sparse CCA applications, one dealing with cross-language document retrieval and the other with vocabulary selection in music annotation. The cross-language document retrieval application involves a collection of documents with each document in different languages, say English and French. The goal is, given a query string in one language, retrieve the most relevant document(s) in the target language. We experimentally show that the proposed sparse CCA algorithm performs similar to the non-sparse version, however using only 10% of non-zero loadings in the canonical components. In the vocabulary selection application, we show that sparse CCA improves the performance of a statistical musical query system by selecting only those words (i.e., pruning the vocabulary) that are correlated to the underlying audio features. In Sect. 4.3, we consider the setting of sparse FDA. By exploiting the special structure of $\boldsymbol{A}$, i.e., $\boldsymbol{A} \in \mathbb{S}^n_+$ with $\operatorname{rank}(\boldsymbol{A}) = 1$, we propose a sparse FDA algorithm that is more efficient than the general sparse GEV algorithm. We also discuss the relation of this algorithm to feature selection in least squares support vector machines (LS-SVMs) (Suykens et al. 2002, Chap. 3), which is a well-studied problem.

In summary, the main contribution of this paper is a globally convergent algorithm (see Algorithm 1)—a sequence of QCQPs—for the sparse GEV problem, which can handle any arbitrary symmetric matrix, $\boldsymbol{A}$ and positive definite matrix, $\boldsymbol{B}$. When $\boldsymbol{A} \in \mathbb{S}^n_+$ and $\boldsymbol{B} \in \mathbb{S}^n_{++}$ is a diagonal matrix, the proposed algorithm has a simple, closed-form update rule. Encouraging empirical results obtained by applying the proposed, general algorithm to some special cases of sparse GEV problems suggest its potential for applications that rely on solving some instance of a sparse GEV problem.

In the following section, we introduce the notation that is used throughout the paper. Supplementary results and related discussions are collected in appendices.

2 Notation

$\mathbb{S}^n$ (resp. $\mathbb{S}^n_+$, $\mathbb{S}^n_{++}$) denotes the set of symmetric (resp. positive semidefinite, positive definite) $n \times n$ matrices defined over $\mathbb{R}$. For $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$, $\boldsymbol{x} \succeq \boldsymbol{0}$ (resp. $\boldsymbol{x} \succ \boldsymbol{0}$) denotes $x_i \geq 0, \forall i$ (resp. $x_i > 0, \forall i$). $\boldsymbol{x} \succeq \boldsymbol{y}$ (resp. $\boldsymbol{x} \preceq \boldsymbol{y}$) denotes $x_i \geq y_i$, $\forall i$ (resp. $x_i \leq y_i$, $\forall i$). $\|\boldsymbol{x}\|_0$ denotes the number of non-zero elements of the vector $\boldsymbol{x}$, $\|\boldsymbol{x}\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$, $1 \leq p < \infty$. $\boldsymbol{I}_n$ denotes an $n \times n$ identity matrix. $\boldsymbol{D}(\boldsymbol{x})$ represents a diagonal matrix formed with $\boldsymbol{x}$ as its principal diagonal.

3 Sparse generalized eigenvalue problem

In the following, we first discuss the intractability of (SGEV-P), which is then used in Sect. 3.1 to motivate and propose a d.c. (difference of convex functions) approximation

to (SGEV-P) that is based on a non-convex approximation to the cardinality constraint. This d.c. program is then solved as a sequence of QCQPs in Sect. 3.3 (see Algorithm 1) using the majorization-minimization method, which is briefly discussed in Sect. 3.2. Since the proposed algorithm is iterative, we present some convergence analysis in Sect. 3.4, which guarantees that the iterates generated by the algorithm converge to a stationary point of the d.c. program.

Intractability of (SGEV-P) Let us consider (SGEV-P), where $\boldsymbol{A} \in \mathbb{S}^n$ and $\boldsymbol{B} \in \mathbb{S}^n_{++}$. Suppose $\boldsymbol{A}$ is not negative definite. Then (SGEV-P) is the maximization of a non-concave objective over the non-convex constraint set $\Phi := \{\boldsymbol{x} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} = 1\} \cap \{\boldsymbol{x} : \|\boldsymbol{x}\|_0 \leq k\}$. Although Φ can be relaxed to a convex set $\widetilde{\Phi} := \{\boldsymbol{x} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1\} \cap \{\boldsymbol{x} : \|\boldsymbol{x}\|_1 \leq k\}$, this does not simplify the problem as the maximization of a non-concave objective over a convex set is NP-hard [p. 342] (Rockafellar 1970).[1] Based on this discussion, it is evident that the intractability of (SGEV-P) is not only due to the cardinality constraint, but also due to the maximization of the non-concave objective function. Therefore, the ℓ_1 approximation that is popularly used in machine learning to obtain a convex approximation to an intractable problem involving a cardinality constraint, is not useful to obtain a tractable approximation to (SGEV-P).

In Appendix A, we discuss a convex approximation to (SGEV-P) using semidefinite programming (SDP) relaxation (Vandenberghe and Boyd 1996). We don't pursue such an approach since it is prohibitively expensive in computation for large n. Instead, in the following section, we trade convexity for good scalability and propose a non-convex approximation to (SGEV-P), resulting in a d.c. program, based on a non-convex approximation to the cardinality constraint.

3.1 Non-convex approximation to $\|\boldsymbol{x}\|_0$ and d.c. formulation

Let us consider the regularized (penalized) version of (SGEV-P) given by

$$\max_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \rho \|\boldsymbol{x}\|_0 : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\}, \tag{SGEV-R}$$

where $\rho > 0$ is the regularization (penalization) parameter. Note that the quadratic equality constraint, $\boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} = 1$ is relaxed to the inequality constraint, $\boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1$. Since

$$\|\boldsymbol{x}\|_0 = \sum_{i=1}^n \mathbb{1}_{\{|x_i| \neq 0\}} = \lim_{\varepsilon \to 0} \sum_{i=1}^n \frac{\log(1 + |x_i|/\varepsilon)}{\log(1 + 1/\varepsilon)}, \tag{5}$$

(SGEV-R) is equivalent (we define two programs to be *equivalent* if their optimizers, i.e., solutions are the same) to

$$\max_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \rho \lim_{\varepsilon \to 0} \sum_{i=1}^n \frac{\log(1 + |x_i|/\varepsilon)}{\log(1 + 1/\varepsilon)} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\}. \tag{6}$$

[1] Note that (GEV-P) also involves the maximization of a non-concave objective over a non-convex set. However, it is well-known that polynomial-time algorithms exist to solve (GEV-P), which is due to its *special* structure of a quadratic objective with a homogeneous quadratic constraint (Boyd and Vandenberghe 2004, p. 229).

The above program is approximated by the following *approximate sparse GEV program* by neglecting the limit in (6) and choosing $\varepsilon > 0$,

$$\max_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \rho \sum_{i=1}^{n} \frac{\log(1+|x_i|/\varepsilon)}{\log(1+1/\varepsilon)} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\}, \tag{7}$$

which is equivalent to

$$\max_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \rho_\varepsilon \sum_{i=1}^{n} \log(|x_i|+\varepsilon) : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\}, \tag{SGEV-A}$$

where $\rho_\varepsilon := \rho/\log(1+\varepsilon^{-1})$. Note that the approximate program in (SGEV-A) is a continuous optimization problem unlike the one in (SGEV-R), which has a combinatorial term. In addition, we show below that the objective of (SGEV-A) exhibits the d.c. structure and therefore can be written as a d.c. program. D.c. programs are well studied and many global optimization algorithms exist to solve them (Horst and Thoai 1999). Before we formulate (SGEV-A) as a d.c. program, we provide the definition of a d.c. program.

Definition 1 (D.c. program, Horst and Thoai 1999) Let Ω be a convex set in $\mathbb{R}^n$. A real valued function $f : \Omega \to \mathbb{R}$ is called a d.c. function on Ω, if there exist two *convex* functions $g, h : \Omega \to \mathbb{R}$ such that f can be expressed in the form $f(\boldsymbol{x}) = g(\boldsymbol{x}) - h(\boldsymbol{x})$, $\boldsymbol{x} \in \Omega$. Optimization problems of the form $\min\{f_0(\boldsymbol{x}) : \boldsymbol{x} \in \Omega, f_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m\}$, where $f_i = g_i - h_i$, $i = 0, \ldots, m$, are d.c. functions are called *d.c. programs*.

To formulate (SGEV-A) as a d.c. program, let us choose $\tau \in \mathbb{R}$ such that $\boldsymbol{A} + \tau \boldsymbol{I}_n \in \mathbb{S}^n_+$. If $\boldsymbol{A} \in \mathbb{S}^n_+$, such τ exists trivially (choose $\tau \geq 0$). If $\boldsymbol{A}$ is indefinite, choosing $\tau \geq -\lambda_{\min}(\boldsymbol{A})$ ensures that $\boldsymbol{A} + \tau \boldsymbol{I}_n \in \mathbb{S}^n_+$. Therefore, choosing $\tau \geq \max(0, -\lambda_{\min}(\boldsymbol{A}))$ ensures that $\boldsymbol{A} + \tau \boldsymbol{I}_n \in \mathbb{S}^n_+$ for any $\boldsymbol{A} \in \mathbb{S}^n$. Equation (SGEV-A) is equivalently written as

$$\min_{\boldsymbol{x}} \left\{ \left[\tau \|\boldsymbol{x}\|_2^2 - \boldsymbol{x}^T(\boldsymbol{A} + \tau \boldsymbol{I}_n)\boldsymbol{x}\right] + \rho_\varepsilon \sum_{i=1}^{n} \log(|x_i|+\varepsilon) : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\}. \tag{8}$$

Introducing the auxiliary variable, $\boldsymbol{y}$, yields the equivalent program

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left\{ \tau \|\boldsymbol{x}\|_2^2 - \left[\boldsymbol{x}^T(\boldsymbol{A} + \tau \boldsymbol{I}_n)\boldsymbol{x} - \rho_\varepsilon \sum_{i=1}^{n} \log(y_i + \varepsilon) \right] : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1,\ -\boldsymbol{y} \preceq \boldsymbol{x} \preceq \boldsymbol{y} \right\}, \tag{9}$$

which is a d.c. program. Indeed, the term $\tau \|\boldsymbol{x}\|_2^2$ is convex in $\boldsymbol{x}$ as $\tau \geq 0$ and, by construction, $\boldsymbol{x}^T(\boldsymbol{A} + \tau \boldsymbol{I}_n)\boldsymbol{x} - \rho_\varepsilon \sum_{i=1}^{n} \log(y_i + \varepsilon)$ is jointly convex in $\boldsymbol{x}$ and $\boldsymbol{y}$. So, the above program is a minimization of the difference of two convex functions over a convex set.

As mentioned before, many global optimization algorithms exist to solve d.c. programs, e.g., branch and bound, cutting planes, etc. However, since these algorithms are not scalable for large n, in Sect. 3.3, we propose a local optimization algorithm based on majorization-minimization (see Sect. 3.2), which allows to exploit the d.c. structure and gives rise to a simple, efficient algorithm: a sequence of QCQPs. Before we discuss the MM method in Sect. 3.2 and present the sparse GEV algorithm in Sect. 3.3, we briefly discuss (a) the approximation to $\|\boldsymbol{x}\|_0$ that we consider in this paper and (b) the behavior of the solution to (SGEV-A)—or equivalently to (9)—as $\varepsilon \to 0$, assuming the global optimum to (SGEV-A) can be obtained.

Approximation to $\|\boldsymbol{x}\|_0$ The approximation (to $\|\boldsymbol{x}\|_0$) that we consider in this paper, i.e.,

$$\|\boldsymbol{x}\|_\varepsilon := \sum_{i=1}^{n} \frac{\log(1+|x_i|\varepsilon^{-1})}{\log(1+\varepsilon^{-1})},$$

has been used in many different contexts: feature selection using SVMs (Weston et al. 2003), sparse signal recovery (Candes et al. 2007), matrix rank minimization (Fazel et al. 2003), etc. This approximation is interesting because of its connection to sparse factorial priors that are studied in Bayesian inference, and can be interpreted as defining a Student's t-distribution prior over $\boldsymbol{x}$, an improper prior given by $\prod_{i=1}^{n} \frac{1}{|x_i|+\varepsilon}$. Tipping (2001) showed that this choice of prior leads to a sparse representation and demonstrated its validity for sparse kernel expansions in the Bayesian framework. Other approximations to $\|\boldsymbol{x}\|_0$ are possible, e.g., Bradley and Mangasarian (1998) used $\sum_{i=1}^{n}(1-e^{-\alpha|x_i|})$ with $\alpha > 0$ ($\|\boldsymbol{x}\|_0 = \lim_{\alpha\to\infty} \sum_{i=1}^{n}(1-e^{-\alpha|x_i|})$) as an approximation to $\|\boldsymbol{x}\|_0$ in the context of feature selection using SVMs.

In Appendix B, we show that the approximation (to $\|\boldsymbol{x}\|_0$) considered in this paper, i.e., $\|\boldsymbol{x}\|_\varepsilon$, is tighter than the ℓ_1-norm approximation, for any $\varepsilon > 0$. Therefore, sparser solutions can be expected for (9)—obtained by replacing $\|\boldsymbol{x}\|_0$ in (SGEV-R) by $\|\boldsymbol{x}\|_\varepsilon$—compared to replacing $\|\boldsymbol{x}\|_0$ by $\|\boldsymbol{x}\|_1$, in (SGEV-R), if the global solution(s) of the non-convex program in (9) can be found.

Behavior of the solution to (SGEV-A) *as* $\varepsilon \to 0$ As mentioned before, (SGEV-A) is an approximation to (SGEV-R), which is obtained by approximating $\|\boldsymbol{x}\|_0$ by $\|\boldsymbol{x}\|_\varepsilon$. Consider the objective functions of (SGEV-R) and (SGEV-A), given as

$$Q(\boldsymbol{x}) := \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \rho\|\boldsymbol{x}\|_0,$$
$$Q_\varepsilon(\boldsymbol{x}) := \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \rho\|\boldsymbol{x}\|_\varepsilon,$$

which are maximized over $\Omega := \{\boldsymbol{x} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \le 1\}$. Suppose that $\widehat{\boldsymbol{x}}$ and $\boldsymbol{x}_\varepsilon$ are solutions to (SGEV-R) and (SGEV-A) respectively, i.e., maximizers of $Q(\boldsymbol{x})$ and $Q_\varepsilon(\boldsymbol{x})$ respectively over Ω. A natural question to answer is whether $\|\boldsymbol{x}_\varepsilon - \widehat{\boldsymbol{x}}\| \to 0$ as $\varepsilon \to 0$? In general, this may not be true because $Q(\boldsymbol{x})$ may be quite flat near its maximum over Ω. If this is not true, at least, one would like to know whether $Q_\varepsilon(\boldsymbol{x}_\varepsilon) \to Q(\widehat{\boldsymbol{x}})$ as $\varepsilon \to 0$, i.e.,

$$\lim_{\varepsilon\to 0} \max_{\boldsymbol{x}\in\Omega} Q_\varepsilon(\boldsymbol{x}) \stackrel{?}{=} \max_{\boldsymbol{x}\in\Omega} Q(\boldsymbol{x}) = \max_{\boldsymbol{x}\in\Omega} \lim_{\varepsilon\to 0} Q_\varepsilon(\boldsymbol{x}). \tag{10}$$

In other words, we would like to know whether the limit process and the maximization over Ω can be interchanged. It can be shown that if Q_ε converges uniformly over Ω to Q, then the equality in (10) holds. However, it is easy to see that Q_ε does not converge uniformly to Q over Ω, so nothing can be said about (10). Therefore, no theoretical guarantees can be provided about the behavior of $Q_\varepsilon(\boldsymbol{x}_\varepsilon)$ w.r.t. $Q(\widehat{\boldsymbol{x}})$ as $\varepsilon \to 0$.

So, from a theoretical point of view, even if we could obtain a global optimum to (SGEV-A), it remains unclear how a global optimum of (SGEV-A) compares to that of (SGEV-R), as $\varepsilon \to 0$. From a practical perspective, as we pointed out, computing a global optimum to (SGEV-A) is computationally hard anyway. Therefore, instead of focusing on a global optimum for (SGEV-A), in Sect. 3.3, we propose a simple and efficient local optimization algorithm for (SGEV-A), based on the MM method. It remains an interesting open problem whether theoretical guarantees can be found, relating a global optimum of (SGEV-A) to that of (SGEV-R), which we will further investigate in future work.

3.2 Majorization-minimization method

The majorization-minimization (MM) method can be thought of as a generalization of the well-known expectation-maximization (EM) algorithm (Dempster et al. 1977). The general principle behind MM algorithms was first enunciated by the numerical analysts, Ortega and Rheinboldt (1970) in the context of line search methods. The MM principle appears in many places in statistical computation, including multidimensional scaling (de Leeuw 1977), robust regression (Huber 1981), correspondence analysis (Heiser 1987), variable selection (Hunter and Li 2005), signal/image processing (Daubechies et al. 2004; Bioucas-Dias et al. 2006; Figueiredo et al. 2007), etc. We refer the interested reader to a tutorial on MM algorithms (Hunter and Lange 2004) and the references therein.

The general idea of MM algorithms is as follows. Suppose we want to minimize f over $\Omega \subset \mathbb{R}^n$. The idea is to construct a *majorization function* g over $\Omega \times \Omega$ such that

$$f(x) \leq g(x, y), \quad \forall x, y \in \Omega \quad \text{and} \quad f(x) = g(x, x), \quad \forall x \in \Omega. \tag{11}$$

Thus, g as a function of x is an upper bound on f and coincides with f at y. The majorization-minimization algorithm corresponding to this majorization function g updates x at iteration l by

$$x^{(l+1)} \in \arg\min_{x \in \Omega} g(x, x^{(l)}), \tag{12}$$

unless we already have

$$x^{(l)} \in \arg\min_{x \in \Omega} g(x, x^{(l)}),$$

in which case the algorithm stops. $x^{(0)}$ is usually chosen randomly. The majorization function, g is usually constructed by using Jensen's inequality for convex functions, the first-order Taylor approximation or the quadratic upper bound principle (Böhning and Lindsay 1988). In fact, any other method can be used to construct g as long as it satisfies (11). It is easy to show that the above iterative scheme decreases the value of f monotonically in each iteration, i.e.,

$$f(x^{(l+1)}) \leq g(x^{(l+1)}, x^{(l)}) \leq g(x^{(l)}, x^{(l)}) = f(x^{(l)}), \tag{13}$$

where the first inequality and the last equality follow from (11) while the sandwiched inequality follows from (12).

Note that MM algorithms can be applied equally well to the maximization of f by simply reversing the inequality sign in (11) and changing the "min" to "max" in (12). In this case, the word MM refers to minorization-maximization, where the function g is called the *minorization function.* To put things in perspective, the EM algorithm can be obtained by constructing the minorization function g using Jensen's inequality for concave functions. The construction of such g is referred to as the E-step, while (12) with the "min" replaced by "max" is referred to as the M-step. The algorithm in (11) and (12) is used in machine learning, e.g., for non-negative matrix factorization (Lee and Seung 2001), under the name *auxiliary function method*. Lange et al. (2000) studied this algorithm under the name *optimization transfer* while Meng (2000) referred to it as the SM algorithm, where "S" stands for the surrogate step (same as the majorization/minorization step) and "M" stands for the minimization/maximization step depending on the problem at hand. g is called the surrogate function.

Having briefly discussed the idea behind MM algorithms, in the following example, we present an MM method to solve d.c. programs, which results in a sequence of convex programs. This will allow to derive an algorithm for the d.c. program in (9).

Example 1 (Linear majorization) Let us consider the optimization problem, $\min\{f(\boldsymbol{x}) : \boldsymbol{x} \in \Omega\}$ where $f = u - v$, with u and v both convex, and v continuously differentiable. Note that, by Definition 1, this is a d.c. program. Now, the goal is to construct an auxiliary function, g. Since v is convex, we have $v(\boldsymbol{x}) \geq v(\boldsymbol{y}) + (\boldsymbol{x} - \boldsymbol{y})^T \nabla v(\boldsymbol{y})$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \Omega$. Therefore,

$$f(\boldsymbol{x}) = u(\boldsymbol{x}) - v(\boldsymbol{x}) \leq u(\boldsymbol{x}) - v(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla v(\boldsymbol{y}) =: g(\boldsymbol{x}, \boldsymbol{y}).$$

It is easy to verify that g is a majorization function of f. Therefore, by (12), we have

$$\boldsymbol{x}^{(l+1)} \in \arg\min_{\boldsymbol{x} \in \Omega} g(\boldsymbol{x}, \boldsymbol{x}^{(l)}) = \arg\min_{\boldsymbol{x} \in \Omega} u(\boldsymbol{x}) - \boldsymbol{x}^T \nabla v(\boldsymbol{x}^{(l)}). \tag{14}$$

If Ω is a convex set, then the above procedure solves a sequence of convex programs. Note that the same idea is used in the concave-convex procedure (CCCP) (Yuille and Rangarajan 2003).

In the following section, we apply the above example to (9) to obtain a sparse GEV algorithm, which is a sequence of QCQPs. We would like to mention that the d.c. program in (9) can be considered as an intermediate step in applying the MM method directly to (SGEV-A), to facilitate the construction of the majorization function. So, it is the general MM framework that essentially allows to derive a simple, efficient local optimization algorithm for the proposed sparse GEV problem formulation.

3.3 Sparse GEV algorithm

Let us return to the approximate sparse GEV program in (9). Let

$$f(\boldsymbol{x}, \boldsymbol{y}) = \tau \|\boldsymbol{x}\|_2^2 + \rho_\varepsilon \sum_{i=1}^{n} \log(\varepsilon + y_i) - \boldsymbol{x}^T (\boldsymbol{A} + \tau \boldsymbol{I}_n) \boldsymbol{x}, \tag{15}$$

where $\tau \geq \max(0, -\lambda_{\min}(\boldsymbol{A}))$ so that (9) can be written as $\min\{f(\boldsymbol{x}, \boldsymbol{y}) : (\boldsymbol{x}, \boldsymbol{y}) \in \Omega\}$ and $\Omega = \{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} \leq 1, \ -\boldsymbol{y} \preceq \boldsymbol{x} \preceq \boldsymbol{y}\}$. Defining

$$u(\boldsymbol{x}, \boldsymbol{y}) := \tau \|\boldsymbol{x}\|_2^2,$$

$$v(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{x}^T (\boldsymbol{A} + \tau \boldsymbol{I}_n) \boldsymbol{x} - \rho_\varepsilon \sum_{i=1}^{n} \log(\varepsilon + y_i),$$

which are convex over $\mathbb{R}^n \times \mathbb{R}^n$ and invoking Example 1, yields the majorization function,

$$\begin{aligned} g((\boldsymbol{x}, \boldsymbol{y}), (\boldsymbol{z}, \boldsymbol{w})) := \tau \|\boldsymbol{x}\|_2^2 - \boldsymbol{z}^T (\boldsymbol{A} + \tau \boldsymbol{I}_n) \boldsymbol{z} + \rho_\varepsilon \sum_{i=1}^{n} \log(\varepsilon + w_i) \\ - 2(\boldsymbol{x} - \boldsymbol{z})^T (\boldsymbol{A} + \tau \boldsymbol{I}_n) \boldsymbol{z} + \rho_\varepsilon \sum_{i=1}^{n} \frac{y_i - w_i}{w_i + \varepsilon}, \end{aligned} \tag{16}$$

and therefore the following algorithm based on (14):

$$\begin{aligned} (\boldsymbol{x}^{(l+1)}, \boldsymbol{y}^{(l+1)}) = \arg\min_{\boldsymbol{x}, \boldsymbol{y}} \quad & \tau \|\boldsymbol{x}\|_2^2 - 2\boldsymbol{x}^T (\boldsymbol{A} + \tau \boldsymbol{I}_n) \boldsymbol{x}^{(l)} + \rho_\varepsilon \sum_{i=1}^{n} \frac{y_i}{y_i^{(l)} + \varepsilon} \\ \text{s.t.} \quad & \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} \leq 1, \quad -\boldsymbol{y} \preceq \boldsymbol{x} \preceq \boldsymbol{y}. \end{aligned} \tag{17}$$

Equation (17) is equivalent to the following *sparse GEV algorithm*,

$$\boldsymbol{x}^{(l+1)} = \arg\min_{\boldsymbol{x}} \left\{ \tau \|\boldsymbol{x}\|_2^2 - 2\boldsymbol{x}^T(\boldsymbol{A} + \tau \boldsymbol{I}_n)\boldsymbol{x}^{(l)} + \rho_\varepsilon \sum_{i=1}^{n} \frac{|x_i|}{|x_i^{(l)}| + \varepsilon} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \le 1 \right\}, \quad \text{(ALG)}$$

which is a sequence of quadratically constrained quadratic programs (QCQPs).[2] We refer to Boyd and Vandenberghe (2004) for details about QCQPs. For $\tau \neq 0$, the optimal solution of (ALG), $\boldsymbol{x}^{(l+1)}$, is unique, since the objective function in (ALG) is strictly convex in $\boldsymbol{x}$ when $\tau \neq 0$. On the other hand, for $\tau = 0$, the objective function is no longer strictly convex. Below, in Special case 1, we show that for $\tau = 0$, (ALG) has a unique optimal solution, which can be found by solving a quadratic program rather than a QCQP.

Note that (ALG) requires the knowledge of τ, which depends on $\lambda_{\min}(\boldsymbol{A})$. Although for certain choices of $\boldsymbol{A}$, $\lambda_{\min}(\boldsymbol{A})$ can be computed efficiently, it is generally an expensive task, which can significantly raise the cost of solving (ALG), if the computation of $\lambda_{\min}(\boldsymbol{A})$ is included as part of solving (ALG). However, if $\boldsymbol{A} \in \mathbb{S}_+^n$ (e.g., for PCA and FDA), τ can be chosen to be zero, to not raise the cost of solving (ALG).

Special case 1 ($\boldsymbol{A} \in \mathbb{S}_+^n$, $\boldsymbol{B} \in \mathbb{S}_{++}^n$, $\tau = 0$) Define $w_i^{(l)} := (|x_i^{(l)}| + \varepsilon)^{-1}$, $\boldsymbol{w}^{(l)} := (w_1^{(l)}, \ldots, w_n^{(l)})$ and $\boldsymbol{W}^{(l)} := \boldsymbol{D}(\boldsymbol{w}^{(l)})$, a diagonal matrix with $\boldsymbol{w}^{(l)}$ as its principal diagonal. Choosing $\tau = 0$ in (ALG), we have that

$$\boldsymbol{x}^{(l+1)} = \arg\max_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}^{(l)} - \frac{\rho_\varepsilon}{2} \left\| \boldsymbol{W}^{(l)}\boldsymbol{x} \right\|_1 : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \le 1 \right\}.$$

We show in Appendix C that when $\rho_\varepsilon > 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$, the unique optimal solution of (ALG) is $\boldsymbol{x}^{(l+1)} = \boldsymbol{0}$. When $\rho_\varepsilon < 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$, we show that

$$\boldsymbol{x}^{(l+1)} = \frac{\boldsymbol{S}(\boldsymbol{S}\boldsymbol{B}\boldsymbol{S})^+(\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})}{\sqrt{(\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})^T(\boldsymbol{S}\boldsymbol{B}\boldsymbol{S})^+(\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})}}, \quad \text{(ALG-R)}$$

where $\boldsymbol{C}^+$ denotes the Moore-Penrose pseudoinverse of $\boldsymbol{C}$, $\boldsymbol{S} := \boldsymbol{D}(\boldsymbol{s})$, $(\boldsymbol{s})_i := \operatorname{sign}((\boldsymbol{A}\boldsymbol{x}^{(l)})_i)$, $(\boldsymbol{\gamma})_i := |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2} w_i^{(l)}$, and

$$\boldsymbol{\lambda}^{(l+1)} \in \arg\min_{\boldsymbol{\lambda}} \left\{ (\boldsymbol{\gamma} + \boldsymbol{\lambda})^T(\boldsymbol{S}\boldsymbol{B}\boldsymbol{S})^+(\boldsymbol{\gamma} + \boldsymbol{\lambda}) : \boldsymbol{\lambda} \succeq \boldsymbol{0} \right\}. \quad (18)$$

The solution to (18) is unique for $(\boldsymbol{\lambda}^{(l+1)})_i$, $i \in \mathcal{I} := \{j : (\boldsymbol{s})_j \neq 0\}$, while it is not unique for $(\boldsymbol{\lambda}^{(l+1)})_i$, $i \notin \mathcal{I}$. However, because of the pre-multiplication by $\boldsymbol{S}$ in (ALG-R), $(\boldsymbol{x}^{(l+1)})_i = 0$, $i \notin \mathcal{I}$ and, therefore, $\boldsymbol{x}^{(l+1)}$ in (ALG-R) is the unique optimal solution of (ALG). Note that in this case of $\tau = 0$, we just need to solve a quadratic program in (18) compared to solving a QCQP in (ALG) when $\tau \neq 0$.

[2] Although (ALG) is not a QCQP in standard form, its equivalent program in (17) can be easily written in standard form as $z^{(l+1)} = \arg\min_z \{z^T \boldsymbol{C} z + \boldsymbol{b}^T z : z^T \boldsymbol{E} z \le 1, z^T \boldsymbol{G} z + \boldsymbol{c}_i^T z \le 0, z^T \boldsymbol{G} z + \boldsymbol{d}_i^T z \le 0, 1 \le i \le n\}$, where $z := [\boldsymbol{x}^T \ \boldsymbol{y}^T]^T$, $\boldsymbol{b} = \begin{bmatrix} -2(\boldsymbol{A}+\tau\boldsymbol{I}_n)\boldsymbol{x}^{(l)} \\ \rho_\varepsilon \boldsymbol{F}^{-1}\boldsymbol{1} \end{bmatrix}$, $\boldsymbol{C} = \begin{bmatrix} \tau\boldsymbol{I}_n & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$, $\boldsymbol{E} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$, $\boldsymbol{F} = \boldsymbol{D}(\boldsymbol{y}^{(l)} + \varepsilon)$ and $\boldsymbol{G}$ is a $2n \times 2n$ matrix of zeroes. $\boldsymbol{c}_i$ and $\boldsymbol{d}_i$ are defined as: for $1 \le i \le n$, $(\boldsymbol{c}_i)_j = \delta_{ij}$, $1 \le j \le n$ and $(\boldsymbol{c}_i)_j = -\delta_{(n+i)j}$, $(n+1) \le j \le 2n$; for $1 \le i \le n$, $(\boldsymbol{d}_i)_j = -\delta_{ij}$, $1 \le j \le n$ and $(\boldsymbol{d}_i)_j = -\delta_{(n+i)j}$, $(n+1) \le j \le 2n$, where δ represents the Kronecker delta. Therefore, for simplicity, we refer to (ALG) as a QCQP.

In case $\rho_\varepsilon = 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$ (rather hypothetical since ρ_ε is a positive *real* number chosen by the user), $\boldsymbol{x}^{(l+1)}$ is no longer unique. However, $\|\boldsymbol{x}^{(l+1)}\|_0$ is usually very small in this case since $(\boldsymbol{x}^{(l+1)})_j = 0$ for $j \notin \arg\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$, as discussed in Appendix C. Also, $\boldsymbol{0}$ is a solution of (ALG) in this case. Therefore, we set $\boldsymbol{x}^{(l+1)} = \boldsymbol{0}$ as the optimal solution of (ALG) when $\rho_\varepsilon \geq 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$, to implement practical algorithms.

Special case 2 ($\boldsymbol{A} \in \mathbb{S}_+^n$, $\boldsymbol{B} \in \mathbb{S}_{++}^n$ diagonal, $\tau = 0$) Let $\boldsymbol{B}_{++}^n$ be a diagonal matrix, i.e., $\boldsymbol{B} = \boldsymbol{D}(\boldsymbol{b})$, where $\boldsymbol{b} = (b_1, \ldots, b_n) \succ \boldsymbol{0}$. Then, it can be shown that (ALG-R) reduces to a simple update rule, given by the closed form expression

$$x_i^{(l+1)} = \frac{[|(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2} w_i^{(l)}]_+ \operatorname{sign}((\boldsymbol{A}\boldsymbol{x}^{(l)})_i)}{b_i \sqrt{\sum_{i=1}^n b_i^{-1} [|(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2} w_i^{(l)}]_+^2}}, \quad \forall i, \tag{ALG-S}$$

where $[a]_+ := \max(0, a)$. See Appendix C for the derivation of (ALG-S). Note that (ALG-S) has a per-iteration complexity of $O(n^2)$ compared to the worst-case complexity of $O(n^3)$ for (ALG) and (ALG-R).

Interpretation of (ALG) Assuming $\tau \neq 0$, (ALG) reduces to

$$\boldsymbol{x}^{(l+1)} = \arg\min_{\boldsymbol{x}} \left\{ \left\| \boldsymbol{x} - (\tau^{-1}\boldsymbol{A} + \boldsymbol{I}_n)\boldsymbol{x}^{(l)} \right\|_2^2 + \frac{\rho_\varepsilon}{\tau} \left\| \boldsymbol{W}^{(l)}\boldsymbol{x} \right\|_1 : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\}. \tag{19}$$

Equation (19) is very similar to LASSO (Tibshirani 1996) except for the *weighted* ℓ_1-penalty and the quadratic constraint. When $\boldsymbol{x}^{(0)}$ is chosen such that $\boldsymbol{x}^{(0)} = a\boldsymbol{1}$, then the first iteration of (19) is a LASSO minimization problem except for the quadratic constraint. Let us analyze (19) to get an intuitive interpretation.

(a) $\rho_\varepsilon = \rho = 0$: (19) reduces to

$$\min_{\boldsymbol{x}} \left\{ \|\boldsymbol{x} - \boldsymbol{t}^{(l)}\|_2^2 : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\},$$

where $\boldsymbol{t}^{(l)} = (\tau^{-1}\boldsymbol{A} + \boldsymbol{I}_n)\boldsymbol{x}^{(l)}$, i.e., the first term in the objective of (19) computes the best approximation to $\boldsymbol{t}^{(l)}$ in the ℓ_2-norm so that the approximation lies in the ellipsoid $\boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1$, and therefore, the solution $\boldsymbol{x}$ is non-sparse.

(b) $\rho_\varepsilon = \rho = \infty$: In this case, (19) reduces to

$$\min_{\boldsymbol{x}} \left\{ \|\boldsymbol{W}^{(l)}\boldsymbol{x}\|_1 : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\},$$

which is a weighted ℓ_1-norm minimization problem. Intuitively, it is clear that if $x_i^{(l)}$ is small, its weighting factor, $w_i^{(l)} = (|x_i^{(l)}| + \varepsilon)^{-1}$ in the next minimization step is large, which therefore pushes $x_i^{(l+1)}$ to be small. This way the small entries in $\boldsymbol{x}$ are generally pushed toward zero as far as the constraints on $\boldsymbol{x}$ allow, therefore yielding a sparse solution.

From the above discussion, it is clear that (19) is a trade-off between the solution to the least-squares problem and the solution to the weighted ℓ_1-norm problem. From now on, we refer to (ALG) as the *Sparse GEV algorithm*, which is detailed in Algorithm 1.

Algorithm 1 Sparse generalized eigenvalue algorithm

Require: $\boldsymbol{A} \in \mathbb{S}^n$, $\boldsymbol{B} \in \mathbb{S}^n_{++}$, $\varepsilon > 0$ and $\rho > 0$

Choose $\tau \geq \max(0, -\lambda_{\min}(\boldsymbol{A}))$

Choose $\boldsymbol{x}^{(0)} \in \{\boldsymbol{x} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1\}$

$\rho_\varepsilon = \frac{\rho}{\log(1+\varepsilon^{-1})}$

if $\tau = 0$ **then**

  if $\boldsymbol{B} = \boldsymbol{D}(\boldsymbol{b})$ **then**

    repeat

      $w_i^{(l)} = (|x_i^{(l)}| + \varepsilon)^{-1}$

      if $\rho_\varepsilon < 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$ **then**

$$x_i^{(l+1)} = \frac{[|(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2} w_i^{(l)}]_+ \operatorname{sign}((\boldsymbol{A}\boldsymbol{x}^{(l)})_i)}{b_i \sqrt{\sum_{i=1}^n b_i^{-1} [|(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2} w_i^{(l)}]_+^2}}, \quad \forall i$$

      else

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{0}$$

      end if

    until convergence

  else

    repeat

      $w_i^{(l)} = (|x_i^{(l)}| + \varepsilon)^{-1}$

      if $\rho_\varepsilon < 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$ **then**

        $(\boldsymbol{\gamma})_i = |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2} w_i^{(l)}$

        $(\boldsymbol{s})_i = \operatorname{sign}((\boldsymbol{A}\boldsymbol{x}^{(l)})_i)$, $\boldsymbol{S} = \boldsymbol{D}(\boldsymbol{s})$

        $\boldsymbol{\lambda}^{(l+1)} \in \arg\min_{\boldsymbol{\lambda}} \left\{ (\boldsymbol{\gamma} + \boldsymbol{\lambda})^T (\boldsymbol{S}\boldsymbol{B}\boldsymbol{S})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda}) : \boldsymbol{\lambda} \succeq \boldsymbol{0} \right\}$

$$\boldsymbol{x}^{(l+1)} = \frac{\boldsymbol{S}(\boldsymbol{S}\boldsymbol{B}\boldsymbol{S})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})}{\sqrt{(\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})^T (\boldsymbol{S}\boldsymbol{B}\boldsymbol{S})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})}}$$

      else

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{0}$$

      end if

    until convergence

  end if

else

  repeat

    $w_i^{(l)} = (|x_i^{(l)}| + \varepsilon)^{-1}$

    $\boldsymbol{W}^{(l)} = \boldsymbol{D}(\boldsymbol{w}^{(l)})$

$$\boldsymbol{x}^{(l+1)} = \arg\min_{\boldsymbol{x}} \left\{ \left\| \boldsymbol{x} - (\tau^{-1}\boldsymbol{A} + \boldsymbol{I}_n)\boldsymbol{x}^{(l)} \right\|_2^2 + \frac{\rho_\varepsilon}{\tau} \left\| \boldsymbol{W}^{(l)} \boldsymbol{x} \right\|_1 : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \leq 1 \right\}$$

  until convergence

end if

return $\boldsymbol{x}^{(l)}$

Choice of ρ, τ and ε To run Algorithm 1, ρ, τ and ε need to be chosen. Based on the above discussion, although it is clear that ρ controls the sparsity of the solution output by Algorithm 1, it is not possible to know a priori the value of ρ that will provide some desired sparsity level. Therefore, in practice, Algorithm 1 is solved for a fixed set of ρ values that are chosen a priori and the solution with desired cardinality is selected. Depending on the specific sparse GEV under consideration, ρ may sometimes be selected in a different way. For example, for supervised learning with FDA, the algorithm can be cross-validated over the set of ρ values that was fixed a priori and the ρ that provides the smallest cross-validation error may be selected. Since ρ is a free parameter, τ and ε can be set to any value (that satisfies the constraints in Algorithm 1) and ρ can be tuned to obtain the desired sparsity as mentioned above. However, it has to be noted that for a fixed value of ρ, increasing τ or ε reduces sparsity.[3] So, in practice τ is chosen to be $\max(0, -\lambda_{\min}(\boldsymbol{A}))$, ε to be *close* to zero and ρ is set by searching for an appropriate value as discussed above.

Post-processing Suppose that Algorithm 1 outputs a solution, $\boldsymbol{x}^*$ such that $\|\boldsymbol{x}^*\|_0 = k$. Can we say that $\boldsymbol{x}^*$ is the optimal solution of (SGEV-P) among all $\boldsymbol{x}$ with cardinality k? The following proposition provides a condition to check for the non-optimality of $\boldsymbol{x}^*$. In addition, it also presents a post-processing step (called *variational renormalization*) that *improves* the performance of Algorithm 1. See Moghaddam et al. (2007b, Proposition 2) for a similar result in the case of $\boldsymbol{A} \in \mathbb{S}_+^n$ and $\boldsymbol{B} = \boldsymbol{I}_n$.

Proposition 1 *Suppose Algorithm 1 converges to a solution $\boldsymbol{x}^*$ such that $\|\boldsymbol{x}^*\|_0 = k$. Let $\boldsymbol{z}$ be the sub-vector of $\boldsymbol{x}^*$ obtained by removing the zero entries of $\boldsymbol{x}^*$ and*

$$\boldsymbol{u}_k = \arg\max_{\boldsymbol{x}} \{\boldsymbol{x}^T \boldsymbol{A}_k \boldsymbol{x} \,:\, \boldsymbol{x}^T \boldsymbol{B}_k \boldsymbol{x} = 1\},$$

where $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ are submatrices of $\boldsymbol{A}$ and $\boldsymbol{B}$ defined by the same non-zero indices of $\boldsymbol{x}^*$. If $\boldsymbol{z} \neq \boldsymbol{u}_k$, then $\boldsymbol{x}^*$ is not the optimal solution of (SGEV-P) *among all $\boldsymbol{x}$ with the same sparsity pattern as $\boldsymbol{x}^*$ (and therefore, is not the optimal solution of* (SGEV-P) *among all $\boldsymbol{x}$ with $\|\boldsymbol{x}\|_0 = k$). Nevertheless, by replacing the non-zero entries of $\boldsymbol{x}^*$ with those of $\boldsymbol{u}_k$, the value of the objective function in* (SGEV-P) *increases from $[\boldsymbol{x}^*]^T \boldsymbol{A} \boldsymbol{x}^*$ to $\lambda(\boldsymbol{A}_k, \boldsymbol{B}_k)$, its optimal value among all $\boldsymbol{x}$ with the same sparsity pattern as $\boldsymbol{x}^*$.*

Proof Assume that $\boldsymbol{x}^*$, the solution output by Algorithm 1, is the optimal solution of (SGEV-P). Define $\boldsymbol{v}$ such that $v_i = \mathbb{1}_{\{|x_i^*| \neq 0\}}$. Since $\boldsymbol{x}^*$ is the optimal solution of (SGEV-P), we have

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{y}} \left\{ \boldsymbol{y}^T \boldsymbol{D}(\boldsymbol{v}) \boldsymbol{A} \boldsymbol{D}(\boldsymbol{v}) \boldsymbol{y} \,:\, \boldsymbol{y}^T \boldsymbol{D}(\boldsymbol{v}) \boldsymbol{B} \boldsymbol{D}(\boldsymbol{v}) \boldsymbol{y} = 1 \right\},$$

which is equivalent to $\boldsymbol{z} = \arg\max\{\boldsymbol{w}^T \boldsymbol{A}_k \boldsymbol{w} : \boldsymbol{w}^T \boldsymbol{B}_k \boldsymbol{w} = 1\} = \boldsymbol{u}_k$ and the result follows. Note that $\lambda(\boldsymbol{A}_k, \boldsymbol{B}_k)$ is the optimal value of (SGEV-P) among all $\boldsymbol{x}$ with the same sparsity pattern as $\boldsymbol{x}^*$. Therefore, if $\boldsymbol{z} = \boldsymbol{u}_k$, then $[\boldsymbol{x}^*]^T \boldsymbol{A} \boldsymbol{x}^* = \lambda(\boldsymbol{A}_k, \boldsymbol{B}_k)$. □

The variational renormalization suggests that given a solution (in our case, $\boldsymbol{x}^*$ at the termination of Algorithm 1), it is almost certainly better to discard the loadings, keep only the

[3] Increasing ε increases the approximation error between $\|\boldsymbol{x}\|_0$ and $\|\boldsymbol{x}\|_\varepsilon$ and therefore reduces sparsity. From (19), it is clear that increasing τ reduces the weight on the term $\|\boldsymbol{W}^{(l)} \boldsymbol{x}\|_1$, which means more importance is given to reducing the approximation error, $\|\boldsymbol{x} - (\tau^{-1} \boldsymbol{A} + \boldsymbol{I}_n) \boldsymbol{x}^{(l)}\|_2^2$, leading to a less sparse solution.

sparsity pattern and solve the smaller unconstrained subproblem to obtain the final loadings, given the sparsity pattern. This procedure surely improves any continuous algorithm's performance.

In Algorithm 1, we mention that the iterative scheme is continued until convergence. What does convergence mean here? Does the algorithm really converge? If it converges, what does it converge to? Does it converge to an optimal solution of (SGEV-A)? To address these questions, in the following section, we provide some convergence analysis of Algorithm 1, using tools from global convergence theory (Zangwill 1969).

3.4 Convergence analysis

For an iterative procedure like Algorithm 1 to be useful, it must converge to point solutions from all or at least a significant number of initialization states and not exhibit other nonlinear system behaviors, such as divergence or oscillation. *Global convergence theory of iterative algorithms* (Zangwill 1969) can be used to investigate this behavior. We mention up front that this *does not* deal with proving convergence to a global optimum. Using this theory, recently, Sriperumbudur and Lanckriet (2009) analyzed the convergence behavior of the iterative linear majorization algorithm in (14) and showed that under certain conditions on u and v, the algorithm in (14) is globally convergent, i.e., for any random initialization, $\boldsymbol{x}^{(0)}$, the sequence of iterates $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ converges to some stationary point—x_* is said to be stationary point of a constrained optimization problem if it satisfies the corresponding Karush-Kuhn-Tucker (KKT) conditions (Bonnans et al. 2006, Sect. 13.3), which assuming constraint qualification are necessary for the local optimality of $\boldsymbol{x}_*$—of the d.c. program, $\min\{u(\boldsymbol{x}) - v(\boldsymbol{x}) : \boldsymbol{x} \in \Omega\}$. Since (ALG) is obtained by applying linear majorization to (9), as shown in Sect. 3.3, the convergence analysis of (ALG) can be carried out by invoking the results in Sriperumbudur and Lanckriet (2009). This results in Theorem 1, which states that Algorithm 1 is globally convergent. This expresses, in a sense, the certainty that the algorithm works. The rate of convergence remains an open problem and is the subject of future work. It is important to stress the fact that global convergence does not imply (contrary to what the term might suggest) convergence to a global optimum for all initial points $\boldsymbol{x}^{(0)}$. Based on Theorem 1, we also present the convergence analysis for some special cases of Algorithm 1 in Corollaries 1–3. The proofs of all these results are provided in Appendix D.

Theorem 1 (Global convergence of sparse GEV algorithm) *Let $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ be any sequence generated by the sparse GEV algorithm in Algorithm 1. Then, all the limit points of $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ are fixed points of Algorithm 1, which are stationary points of the program in* (SGEV-A),

$$\rho_\varepsilon \sum_{i=1}^{n} \log(\varepsilon + |x_i^{(l)}|) - [\boldsymbol{x}^{(l)}]^T \boldsymbol{A}\boldsymbol{x}^{(l)} \to \rho_\varepsilon \sum_{i=1}^{n} \log(\varepsilon + |x_i^*|) - [\boldsymbol{x}^*]^T \boldsymbol{A}\boldsymbol{x}^* := L^*, \qquad (20)$$

for some fixed point $\boldsymbol{x}^*$, $\|\boldsymbol{x}^{(l+1)} - \boldsymbol{x}^{(l)}\| \to 0$, and either $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ converges or the set of limit points of $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ is a connected and compact subset of $\mathscr{S}(L^*)$, where $\mathscr{S}(a) := \{\boldsymbol{x} \in \mathscr{S} : \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \rho_\varepsilon \sum_{i=1}^{n} \log(\varepsilon + |x_i|) = -a\}$ and $\mathscr{S}$ is the set of fixed points of Algorithm 1. If $\mathscr{S}(L^*)$ is finite, then any sequence $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ generated by Algorithm 1 converges to some $\boldsymbol{x}^*$ in $\mathscr{S}(L^*)$.

Having considered the convergence of Algorithm 1, we now consider the convergence of some of its special cases. The following result shows that a simple iterative algorithm can be obtained to compute the generalized eigenvector associated with $\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B})$, when $\boldsymbol{A} \in \mathbb{S}_+^n$.

Corollary 1 *Let* $\boldsymbol{A} \in \mathbb{S}^n_+$, $\tau = 0$ *and* $\rho = 0$. *Then, any sequence* $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ *generated by the following algorithm*

$$\boldsymbol{x}^{(l+1)} = \frac{\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}^{(l)}}{\sqrt{[\boldsymbol{x}^{(l)}]^T\boldsymbol{A}\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}^{(l)}}}, \tag{21}$$

converges to some $\boldsymbol{x}^*$ *such that* $\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B}) = [\boldsymbol{x}^*]^T\boldsymbol{A}\boldsymbol{x}^*$ *and* $[\boldsymbol{x}^*]^T\boldsymbol{B}\boldsymbol{x}^* = 1$.

Assuming $\boldsymbol{B} = \boldsymbol{I}_n$, $\rho = 0$ and $\tau = 0$, as a corollary to Corollary 1, the following result shows that Algorithm 1 reduces to the power method for computing $\lambda_{\max}(\boldsymbol{A})$, and the sequence of iterates generated by Algorithm 1 converges to the eigenvector associated with $\lambda_{\max}(\boldsymbol{A})$.

Corollary 2 (Power method) *Let* $\boldsymbol{A} \in \mathbb{S}^n_+$, $\boldsymbol{B} = \boldsymbol{I}_n$, $\tau = 0$ *and* $\rho = 0$. *Then Algorithm 1 is the power method for computing* $\lambda_{\max}(\boldsymbol{A})$, *wherein any sequence* $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ *generated by Algorithm 1 converges to some* $\boldsymbol{x}^* \in \{\boldsymbol{x} : \|\boldsymbol{x}\|_2^2 = 1\}$ *such that* $\lambda_{\max}(\boldsymbol{A}) = [\boldsymbol{x}^*]^T\boldsymbol{A}\boldsymbol{x}^*$.

Finally, Corollary 3 generalizes the above result for any $\boldsymbol{A} \in \mathbb{S}^n$ with $\lambda_{\max}(\boldsymbol{A}) > 0$, i.e., the sequence of iterates generated by Algorithm 1 converges to the eigenvector associated with $\lambda_{\max}(\boldsymbol{A})$ for any $\boldsymbol{A} \in \mathbb{S}^n$ with $\lambda_{\max}(\boldsymbol{A}) > 0$, and not just for $\boldsymbol{A} \in \mathbb{S}^n_+$.

Corollary 3 *Let* $\boldsymbol{A} \in \mathbb{S}^n$ *such that* $\lambda_{\max}(\boldsymbol{A}) > 0$. *Assume* $\boldsymbol{B} = \boldsymbol{I}_n$ *and* $\rho = 0$. *Then, any sequence* $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ *generated by Algorithm 1 converges to some* $\boldsymbol{x}^* \in \{\boldsymbol{x} : \|\boldsymbol{x}\|_2^2 = 1\}$ *such that* $\lambda_{\max}(\boldsymbol{A}) = [\boldsymbol{x}^*]^T\boldsymbol{A}\boldsymbol{x}^*$.

In summary, in this section, we have presented our main contribution: a globally convergent sparse GEV algorithm (Algorithm 1) which can handle any $\boldsymbol{A} \in \mathbb{S}^n$ and $\boldsymbol{B} \in \mathbb{S}^n_{++}$. In the following section, we illustrate the potential of this general algorithm for some specific instances of sparse GEV problems, corresponding to specific choices for $\boldsymbol{A}$ and $\boldsymbol{B}$: sparse PCA, sparse CCA and sparse FDA.

4 Experimental results

To illustrate the potential of the sparse GEV algorithm (Algorithm 1), suited for any application that involves a specific instance of a sparse GEV problem, we present empirical results for some of its special cases in this section. To this end, in Sect. 4.1, we choose $\boldsymbol{A} \in \mathbb{S}^n_+$, $\boldsymbol{B} = \boldsymbol{I}_n$ and $\tau = 0$—a special case of (ALG-S)—to obtain a sparse PCA algorithm, called DC-PCA, which is then compared to various other sparse PCA algorithms (that are proposed in literature) on both benchmark and three high-dimensional gene datasets. Experiments show that the performance of DC-PCA in terms of explained variance vs. cardinality, as well as its computational efficiency compare well to the state-of-the-art amongst these other algorithms. In Sect. 4.2, we choose $\boldsymbol{A} \in \mathbb{S}^n$ (with $\boldsymbol{A}$ being indefinite as shown beneath (3)) and $\boldsymbol{B} \in \mathbb{S}^n_{++}$ to obtain a sparse CCA algorithm, which is evaluated for two sparse CCA applications: cross-language document retrieval and vocabulary selection in music annotation. In Sect. 4.3, we choose $\boldsymbol{A} \in \mathbb{S}^n_+$ with $\operatorname{rank}(\boldsymbol{A}) = 1$ and $\boldsymbol{B} \in \mathbb{S}^n_{++}$, which is the setting of FDA (see (4)). Exploiting the special structure of $\boldsymbol{A}$ for FDA allows to propose a sparse FDA algorithm that is more efficient than the general sparse GEV algorithm in Algorithm 1. We also discuss its relation to feature selection in least squares SVMs (Suykens et al. 2002, Chap. 3).

Algorithm 2 Sparse PCA algorithm (DC-PCA)

Require: $\boldsymbol{A} \in \mathbb{S}_+^n$, $\varepsilon > 0$ and $\rho > 0$

Choose $\boldsymbol{x}^{(0)} \in \{\boldsymbol{x} : \boldsymbol{x}^T\boldsymbol{x} \leq 1\}$

$\rho_\varepsilon = \frac{\rho}{\log(1+\varepsilon^{-1})}$

repeat

$w_i^{(l)} = (|x_i^{(l)}| + \varepsilon)^{-1}$

if $\rho_\varepsilon < 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$ **then**

$$x_i^{(l+1)} = \frac{[|(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2} w_i^{(l)}]_+ \operatorname{sign}((\boldsymbol{A}\boldsymbol{x}^{(l)})_i)}{\sqrt{\sum_{i=1}^n [|(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2} w_i^{(l)}]_+^2}}, \quad \forall i$$

else

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{0}$$

end if

until convergence

return $\boldsymbol{x}^{(l)}$

4.1 Sparse principal component analysis

In this section, we consider sparse PCA as a special case of the sparse GEV algorithm, presented in Sect. 3.3, for $\boldsymbol{A} \in \mathbb{S}_+^n$ being a covariance matrix, $\boldsymbol{B} = \boldsymbol{I}_n$ and $\tau = 0$, and call the resulting algorithm DC-PCA (see Algorithm 2). Note that this is a special case of (ALG-S) with $b_i = 1$, $\forall i$. Since the computation of $\boldsymbol{x}^{(l+1)}$, from $\boldsymbol{x}^{(l)}$, involves computing $\boldsymbol{A}\boldsymbol{x}^{(l)}$, which has a complexity of $O(n^2)$, the DC-PCA algorithm has a per iteration complexity of $O(n^2)$. Being a special case of Algorithm 1, which is a globally convergent algorithm, DC-PCA is also globally convergent.

Before empirically comparing the performance of DC-PCA to other sparse PCA algorithms that have been proposed in literature, we briefly discuss the prior work on sparse PCA algorithms. The earliest attempts at "sparsifying" PCA consisted of simple axis rotations and component thresholding (Cadima and Jolliffe 1995) for subset selection, often based on the identification of principal variables (McCabe 1984). The first true computational technique, called SCoTLASS (Jolliffe et al. 2003), provided an optimization framework using LASSO (Tibshirani 1996) by enforcing a sparsity constraint on the PCA solution by bounding its ℓ_1-norm, leading to a non-convex procedure. Zou et al. (2006) proposed a ℓ_1-penalized regression algorithm for PCA (called SPCA) using an *elastic net* (Zou and Hastie 2005) and solved it efficiently using least angle regression (Efron et al. 2004). Subsequently, d'Aspremont et al. (2007) proposed a convex relaxation to the non-convex cardinality constraint for PCA (called DSPCA) leading to a SDP with a complexity of $O(n^4\sqrt{\log n})$. Although this method shows performance comparable to SPCA on a small-scale benchmark data set, it is not scalable for high-dimensional data sets, even possibly with Nesterov's first-order method (Nesterov 2005). Moghaddam et al. (2007b) proposed a combinatorial optimization algorithm (called GSPCA) using greedy search and branch-and-bound methods to solve the sparse PCA problem, leading to a total complexity of $O(n^4)$ for a full set of solutions (one for each target sparsity between 1 and n). d'Aspremont et al. (2008) formulated a new SDP relaxation to the sparse PCA problem and derived a more efficient greedy algorithm (compared to GSPCA) for computing a full set of solutions at a total numerical

complexity of $O(n^3)$, which is based on the convexity of the largest eigenvalue of a symmetric matrix. Recently, Journée et al. (2010) proposed a simple, iterative sparse PCA algorithm (GPower$_{\ell_0}$) with a per iteration complexity of $O(n^2)$ and showed that it performs similar or better than many of the above mentioned algorithms.

We now illustrate the effectiveness of DC-PCA in terms of sparsity and scalability on various datasets by comparing to different approaches. On small datasets, the performance of DC-PCA is compared against SPCA, DSPCA, GSPCA and GPower$_{\ell_0}$, while on large datasets, DC-PCA is compared to all these algorithms except DSPCA and GSPCA due to scalability issues. Since GPower$_{\ell_0}$ has been compared to the greedy algorithm of d'Aspremont et al. (2008) by Journée et al. (2010), wherein it is shown that these two algorithms perform similarly except for the greedy algorithm being computationally more complex, we do not include the greedy algorithm in our comparison. The results show that the performance of DC-PCA is comparable to the performance of many of these algorithms, with state-of-the-art *scalability*, i.e., comparable to GPower$_{\ell_0}$. This competitive comparison to the state-of-the-art for the special case of sparse PCA, which is a well-studied problem, illustrates the potential of the general sparse GEV algorithm, proposed in this work, for other instances of sparse GEV problems that have not been studied extensively. The experiments in this paper are carried out on a Linux 3 GHz, 4 GB RAM workstation. On the implementation side, we fix ε to be the machine precision in all our experiments, which is motivated from the discussion in Sect. 3.1.

4.1.1 Pit props data

The pit props dataset (Jeffers 1967) has become a standard benchmark example to test sparse PCA algorithms. The first 6 principal components (PCs) capture 87% of the total variance. Therefore, the explanatory power of sparse PCA methods is often compared on the first 6 sparse PCs.[4] Table 1 shows the first 3 sparse PCs and their loadings for SPCA, DSPCA, GSPCA, GPower$_{\ell_0}$ and DC-PCA. Using the first 6 sparse PCs, SPCA captures 75.8% of the variance with a cardinality pattern of $(7, 4, 4, 1, 1, 1)$, which indicates the number of non-zero loadings for the first to the sixth sparse PC, respectively. This results in a total of 18 non-zero loadings for SPCA, while DSPCA captures 75.5% of the variance with a sparsity pattern of $(6, 2, 3, 1, 1, 1)$, totaling 14 non-zero loadings. With a sparsity pattern of $(6, 2, 2, 1, 1, 1)$ (total of *only* 13 non-zero loadings), DC-PCA, GSPCA and GPower$_{\ell_0}$ can capture 77.1% of the total variance. Comparing the cumulative variance and cumulative cardinality, Figs. 1(a–b) show that DC-PCA explains more variance with fewer non-zero loadings than SPCA and DSPCA. In addition, its performance is similar to that of GSPCA and GPower$_{\ell_0}$. For the first sparse PC, Fig. 1(c) shows that DC-PCA consistently explains more variance with better sparsity than SPCA, while performing similar to other algorithms. Figure 1(d) shows the variation of sparsity and explained variance with respect to ρ for the first sparse PC computed with DC-PCA. This plot summarizes the method for setting ρ: the algorithm is run for various ρ and the value of ρ that achieves the desired sparsity is selected.

[4]The discussion so far dealt with computing the first sparse eigenvector (which we also call the "first sparse PC") of $\boldsymbol{A}$. To compute the second sparse eigenvector (or "second sparse PC"), the matrix $\boldsymbol{A}$ is deflated with the first sparse eigenvector (see Mackey 2009 for details) and the sparse PCA algorithm is applied again. In general, subsequent sparse eigenvectors (or "sparse PCs") are obtained by applying the sparse PCA algorithm to a sequence of deflated matrices. In this paper, we use the orthogonalized Hotelling's deflation, as mentioned in Mackey (2009). More details can also be found in Sriperumbudur et al. (2007).

Table 1 Loadings for first three sparse principal components (PCs) of the pit props data. The SPCA and DSPCA loadings are taken from Zou et al. (2006) and d'Aspremont et al. (2007) respectively

	PC	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}
SPCA	1	−.48	−.48	0	0	.18	0	−.25	−.34	−.42	−.40	0	0	0
	2	0	0	.79	.62	0	0	0	−.02	0	0	0	.01	0
	3	0	0	0	0	.64	.59	.49	0	0	0	0	0	−.02
DSPCA	1	−.56	−.58	0	0	0	0	−.26	−.10	−.37	−.36	0	0	0
	2	0	0	.71	.71	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	−.79	−.61	0	0	0	0	0	.01
GSPCA	1	.44	.45	0	0	0	0	.38	.34	.40	.42	0	0	0
	2	0	0	.71	.71	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	.82	.58	0	0	0	0	0	0
GPower$_{\ell_0}$	1	.44	.45	0	0	0	0	.38	.34	.40	.42	0	0	0
	2	0	0	.71	.71	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	.82	.58	0	0	0	0	0	0
DC-PCA	1	.45	.46	0	0	0	0	.37	.33	.40	.42	0	0	0
	2	0	0	.71	.71	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	.82	.58	0	0	0	0	0	0

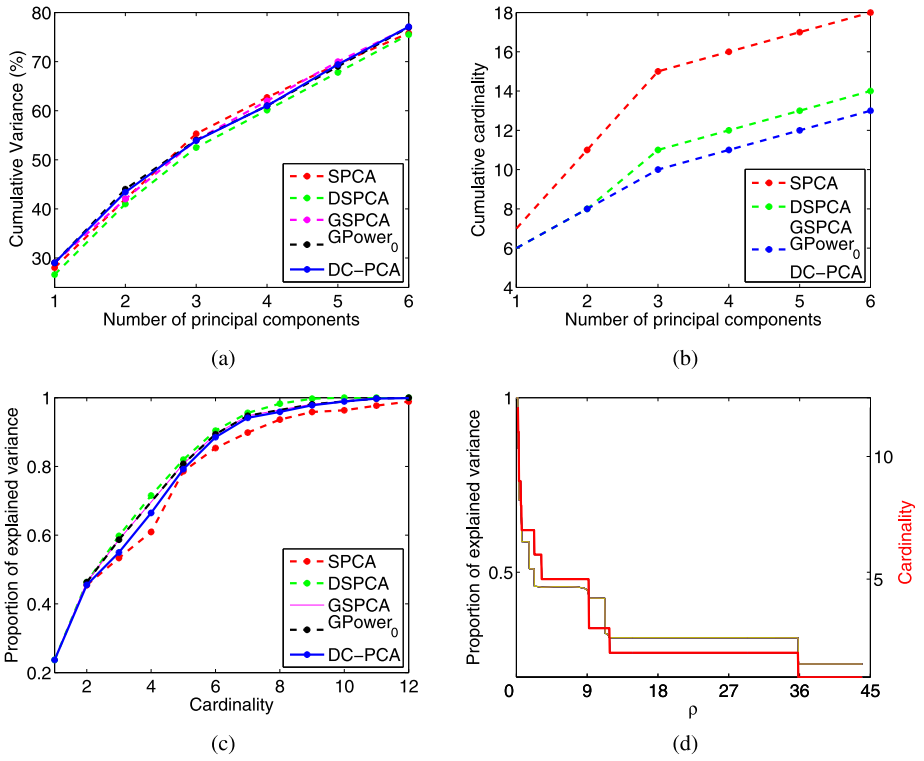

Fig. 1 (Color online) Pit props: (**a**) cumulative variance and (**b**) cumulative cardinality for the first 6 sparse principal components (PCs); (**c**) proportion of explained variance (PEV) vs. cardinality for the first sparse PC (obtained by varying the sparsity parameter and computing the cardinality and explained variance for the solution vector); (**d**) dependence of sparsity (shown in *red*) and PEV (shown in *brown*) on ρ for the first sparse PC computed with DC-PCA

4.1.2 Random test problems

In this section, we follow the experimental setup that is considered in Journée et al. (2010). Throughout this section, we assume $\boldsymbol{A} = \boldsymbol{C}^T\boldsymbol{C}$, where $\boldsymbol{C}$ is a $p \times n$ random matrix whose entries are generated according to a Gaussian distribution, with zero mean and unit variance. In the following, we present the trade-off curves (proportion of explained variance vs. cardinality for the first sparse PC associated with $\boldsymbol{A}$), computational complexity vs. cardinality and computational complexity vs. problem size for various sparse PCA algorithms.

Trade-off curves Figure 2(a) shows the trade-off between the proportion of explained variance and the cardinality for the first sparse PC associated with $\boldsymbol{A}$ for various sparse PCA algorithms. For each algorithm, the sparsity inducing parameter (k in the case of DSPCA and GSPCA, and the regularization parameter in the case of SPCA, GPower$_{\ell_0}$ and DC-PCA) is incrementally increased to obtain the solution vector with cardinality that decreases from n to 1. The results displayed in Fig. 2(a) are averages of computations on 100 random matrices with dimensions $p = 100$ and $n = 300$. It can be seen from Fig. 2(a) that DC-PCA performs similar to DSPCA, GSPCA and GPower$_{\ell_0}$, while performing better than SPCA.

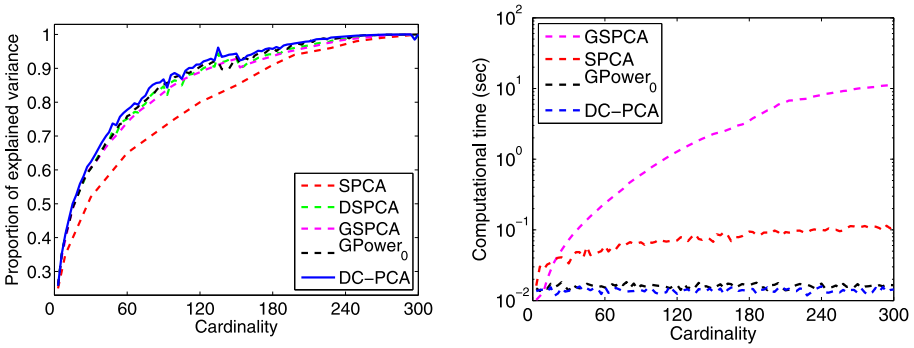

Fig. 2 Random test data: (**a**) (average) proportion of explained variance vs. cardinality for the first sparse PC of $\boldsymbol{A}$; (**b**) (average) computation time vs. cardinality. In (**a**), all the sparse PCA algorithms perform similarly and better than SPCA. In (**b**), the complexity of GSPCA grows significantly with increasing cardinality of the solution vector, while the speed of the other methods is almost independent of the cardinality

Table 2 Average computation time (in seconds) for the first sparse PC associated with $\boldsymbol{A}$ for a fixed regularization parameter

$p \times n$	100×1000	250×2500	500×5000	750×7500	1000×10000
SPCA	0.135	1.895	10.256	34.367	87.459
GPower_{ℓ_0}	0.027	0.159	0.310	1.224	1.904
DC-PCA	0.034	0.151	0.301	1.202	1.913

Computational complexity vs. cardinality Figure 2(b) shows the average time required by the sparse PCA algorithms to extract the first sparse PC of $\boldsymbol{A}$ with $p = 100$ and $n = 300$, for varying cardinality. It is obvious from Fig. 2(b) that as the cardinality increases, GSPCA tends to get slower while the speed of SPCA, GPower_{ℓ_0} and DC-PCA is not really affected by the cardinality. We did not show the results of DSPCA in Fig. 2(b) as its computational complexity is an order of magnitude (around 100 times) more than the scale on the vertical axis of Fig. 2(b). Journée et al. (2010) have demonstrated that the greedy method proposed by d'Aspremont et al. (2008) also exhibits the behavior of increasing computational complexity with the increase in cardinality.

Computational complexity vs. problem size Figure 3 shows the average computation time in seconds, required by various sparse PCA algorithms, to extract the first sparse PC of $\boldsymbol{A}$, for various problem sizes, n, where n is increased exponentially and p is fixed to 500. The times shown are averages over 100 random instances of $\boldsymbol{A}$ for each problem size, where the sparsity inducing parameters are chosen such that the solution vectors of these algorithms exhibit comparable cardinality. It is clear from Fig. 3 that DC-PCA and GPower_{ℓ_0} scale better to large-dimensional problems than the other algorithms. Since, on average, GSPCA and DSPCA are much slower than the other methods, even for low cardinalities (see Fig. 2(b)), we discard them from all the following numerical experiments that deal with large n.

For the remaining algorithms, SPCA, GPower_{ℓ_0} and DC-PCA, we run another round of experiments, now examining the computational complexity with varying n and p but with a fixed aspect ratio $n/p = 10$. The results are depicted in Table 2. Again, the corresponding

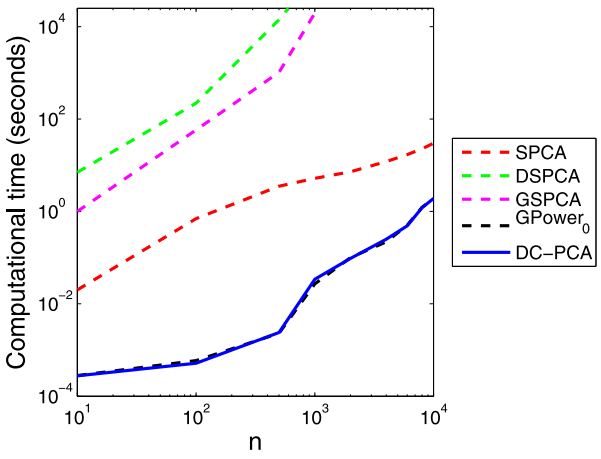

Fig. 3 Average computation time (seconds) for the first sparse PC of $\boldsymbol{A}$ vs. problem size, n, over 100 randomly generated matrices $\boldsymbol{A}$

Table 3 Gene expression datasets

Dataset	Samples (p)	Genes (n)	Reference
Colon cancer	62	2000	Alon et al. (1999)
Leukemia	38	7129	Golub et al. (1999)
Ramaswamy	127	16063	Ramaswamy et al. (2001)

regularization parameters are set in such a way that the solution vectors of these algorithms exhibit comparable cardinality. The values displayed in Table 2 correspond to the average running times of the algorithms on 100 random instances of $\boldsymbol{A}$ for each problem size. It can be seen that our proposed method, DC-PCA, is comparable to GPower$_{\ell_0}$ and faster than SPCA.

4.1.3 Gene expression data

Gene expression data from DNA microarrays provides the expression level of thousands of genes across several hundreds or thousands of experiments. To enhance the interpretation of these large data sets, sparse PCA algorithms can be applied, to extract sparse principal components that involve only a few genes.

Datasets Usually, gene expression data is specified by a $p \times n$ matrix (say $\boldsymbol{C}$) of p samples and n genes. The covariance matrix, $\boldsymbol{A}$ is therefore computed as $\boldsymbol{C}^T\boldsymbol{C}$. In our experiments, we consider three gene expression datasets which are tabulated in Table 3. The colon cancer dataset (Alon et al. 1999) consists of 62 tissue samples (22 normal and 40 cancerous) with the gene expression profiles of $n = 2000$ genes extracted from DNA microarray data. Its first principal component explains 44.96% of the total variance. The leukemia dataset (Golub et al. 1999) consists of a training set of 38 samples (27 ALL and 11 AML, two variants of leukemia) from bone marrow specimens and a test set of 34 samples (20 ALL and 14 AML). This dataset has been used widely in classification settings where the goal is to distinguish between two variants of leukemia. All samples have 7129 features, each of which corresponds to a normalized gene expression value extracted from the microarray image. The

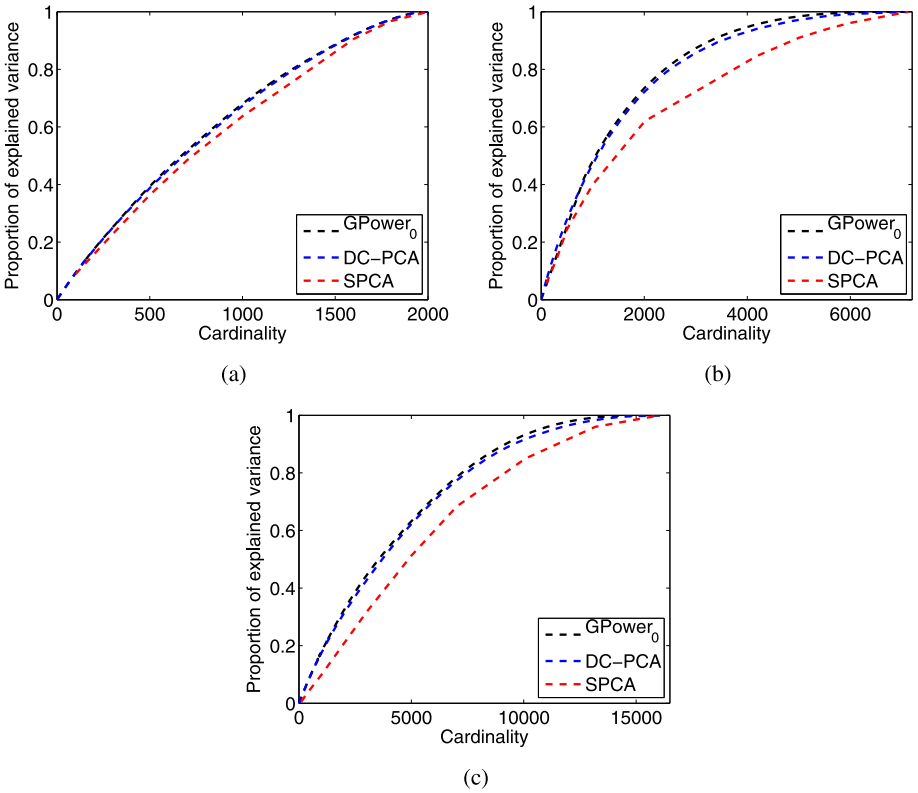

Fig. 4 Trade-off curves between explained variance and cardinality for (**a**) colon cancer, (**b**) leukemia and (**c**) Ramaswamy datasets. The proportion of variance explained is computed on the first sparse principal component. (**a–c**) show that DC-PCA performs similar to GPower$_{\ell_0}$, while explaining more variance (for a fixed cardinality) than SPCA

first principal component explains 87.64% of the total variance. The Ramaswamy dataset has 16063 genes and 127 samples, its first principal component explaining 76.5% of the total variance.

The high dimensionality of these datasets makes them suitable candidates for studying the performance of sparse PCA algorithms, by investigating their ability to explain variance in the data based on a small number of genes, to obtain interpretable results. Since DSPCA and GSPCA are not scalable for these large datasets, in our study, we compare DC-PCA to SPCA and GPower$_{\ell_0}$.

Trade-off curves Figures 4(a–c) show the proportion of explained variance versus the cardinality for the first sparse PC for the datasets shown in Table 3. It can be seen that DC-PCA performs similar to GPower$_{\ell_0}$ and performs better than SPCA.

Computational complexity The average computation time required by the sparse PCA algorithms on each dataset is shown in Table 4. The indicated times are averages over n computations, one for each cardinality ranging from n down to 1. The results show that DC-PCA and GPower$_{\ell_0}$ are significantly faster than SPCA, which, for a long time, was widely accepted as the algorithm that can handle large datasets.

Table 4 Computation time (in seconds) to obtain the first sparse PC, averaged over cardinalities ranging from 1 to n, for the colon cancer, leukemia and Ramaswamy datasets

	Colon cancer	Leukemia	Ramaswamy
n	2000	7129	16063
SPCA	2.057	3.548	38.731
GPower$_{\ell_0}$	0.182	0.223	2.337
DC-PCA	0.034	0.156	0.547

Overall, the results in this section demonstrate that DC-PCA performs similar to or better than various sparse PCA algorithms proposed in literature, both in terms of scalability and proportion of variance explained vs. cardinality. As mentioned before, DC-PCA (see Algorithm 2) is a special instance of a more general framework (see Algorithm 1), that can be used to address other sparse generalized eigenvalue problems as well, e.g., sparse CCA, sparse FDA, etc., whereas most other sparse PCA algorithms cannot be readily extended to these other settings.

4.2 Sparse canonical correlation analysis

In this section, we consider sparse CCA as a special case of the sparse GEV algorithm and present two CCA applications where sparsity is helpful. We call our sparse CCA algorithm DC-CCA, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are determined from the covariance and cross-covariance matrices as explained right below (3). Note that $\boldsymbol{A}$ is indefinite and, therefore, in our experiments, we choose $\tau = -\lambda_{\min}(\boldsymbol{A})$ in Algorithm 1. In the following, we present two sparse CCA applications, one related to the task of cross-language document retrieval and the other dealing with semantic annotation and retrieval of music (Torres et al. 2007a, 2007b).

4.2.1 Cross-language document retrieval

The problem of cross-language document retrieval involves a collection, $\{D_i\}_{i=1}^N$ of documents, with each document being represented in different languages, say English and French. The goal of the task is, given a query string in one language, retrieve the most relevant document(s) in the target language. The first step is to obtain a semantic representation of the documents in both languages, which models the correlation between translated versions, so we can detect similarities in content between the two document spaces (one for English and the other for French). This is exactly what CCA does by finding a low-dimensional representation in both languages, with maximal correlation between them. Vinokourov et al. (2003) used CCA to address this problem and showed that the CCA approach performs better than the latent semantic indexing approach used by Littman et al. (1998). CCA provides an *efficient* basis representation (that captures the maximal correlation) for the two document spaces.

Using a bag-of-words representation for the documents, sparse CCA would allow to find a low-dimensional model based on a small subset of words in both languages. This would improve the interpretability of the model and could identify small subsets of words that are used in similar contexts in both languages and, possibly, are translations of one another. Representing documents by their similarity to all other documents (e.g., by taking inner products

of bag-of-word vectors, as explained below), sparse CCA would create a low-dimensional model that only requires to measure the similarity for a small subset of the training documents. This would immediately improve storage requirements and the efficiency of retrieval computations. In this study, we follow the second approach, representing documents by their similarity to all other training documents by applying a linear kernel function to a binary bag-of-words representation of the documents, as proposed in Vinokourov et al. (2003). This will illustrate how we can achieve significant sparsity without significant loss of retrieval performance.

More specifically, each version of a document (English or French) is modeled using a bag-of-words feature vector. Within a feature vector, we associate an element in $\{0, 1\}$ with each word w_i in its language vocabulary. A value of 1 indicates that w_i is found in the document. We collect the feature vectors into the $n \times p$ matrix $\boldsymbol{E}$, where we collect the English feature vectors, and the $n \times q$ matrix $\boldsymbol{F}$, where we collect the French feature vectors. n is the number of documents and p and q are the vocabulary sizes of $\boldsymbol{E}$ and $\boldsymbol{F}$ respectively. Computing the similarity between English documents as the inner product between their binary bag-of-words vectors (i.e., the rows of $\boldsymbol{E}$) results in computing an $n \times n$ data matrix $\boldsymbol{E}\boldsymbol{E}^T$. Similarly, we compute an $n \times n$ data matrix $\boldsymbol{F}\boldsymbol{F}^T$ and obtain two feature spaces which are both n-dimensional.

By applying sparse CCA, we effectively perform simultaneous feature selection across two vector spaces and characterize the content of and correlation between English and French documents in an efficient manner. We use the DC-CCA algorithm, using the covariance and cross-variance matrices associated with the document matrices $\boldsymbol{E}\boldsymbol{E}^T$ and $\boldsymbol{F}\boldsymbol{F}^T$ and obtain successive pairs of sparse canonical components which we stack into the columns of $\boldsymbol{V}_E$ and $\boldsymbol{V}_F$. Subsequent pairs of these sparse canonical components are obtained by deflating $\boldsymbol{E}\boldsymbol{E}^T$ and $\boldsymbol{F}\boldsymbol{F}^T$ with respect to previous canonical components. For a detailed review on deflation, we refer the reader to Shawe-Taylor and Cristianini (2004).

Then, given a query document in an input language, say English, we convert the query into the appropriate feature vector, $\boldsymbol{q}_E$. We project $\boldsymbol{q}_E$ onto the subspace spanned by the sparse canonical components in the English language space by computing $\boldsymbol{V}_E^T\boldsymbol{q}_E$.[5] Similarly, we project all the French documents onto the subspace spanned by the sparse canonical components, $\boldsymbol{V}_F$ associated with the French language. Finally, we perform document retrieval by selecting those French documents whose projections are closest to the projected query, where we measure distance in a nearest neighbor sense.

Experimental details The data set used was the Aligned Hansards of the 36th Parliament of Canada (Germann 2001), which is a collection of 1.3 million pairs of text chunks (sentences or smaller fragments) aligned into English and French translations. The text chunks are split into documents based on $***$ delimiters. After removing stop words and rare words (those that occur less than 3 times), we are left with an 1800×26328 English document-by-term matrix and a 1800×30167 French matrix. Computing $\boldsymbol{E}\boldsymbol{E}^T$ and $\boldsymbol{F}\boldsymbol{F}^T$ results in matrices of size 1800×1800.

To generate a query, we select English test documents from a test set not used for training. The appropriate retrieval result is the corresponding French language version of the query document. To perform retrieval, the query and the French test documents are projected onto the sparse canonical components and retrieval is performed as described before. Table 5 shows the performance of DC-CCA (sparse CCA) against CCA. We measure our

[5]Notice how this projection, onto the sparse canonical components, only requires to compute a few elements of $\boldsymbol{q}_E$, i.e., the ones corresponding to the non-zero loadings of the sparse canonical components; differently said, we only need to compute the similarity of the query document to a small subset of all training documents.

Table 5 Average area under the ROC curve (in %) using CCA and sparse CCA (DC-CCA) in a cross-language document retrieval task. *d* represents the number of canonical components and *sparsity* represents the percentage of zero loadings in the canonical components

d	100	200	300	400	500
CCA	99.92	99.93	99.96	99.95	99.93
DC-CCA	95.72	97.57	98.45	98.75	99.04
Sparsity	87.15	87.56	87.95	88.21	88.44

results using the average area under the ROC curve (average AROC). The results in Table 5 are shown in percentages. To go into detail, for each test query we generate an ROC curve from the ranked retrieval results. Results are ranked according to their projected feature vector's Euclidean distance from the query. The area under this ROC curve is used to measure performance. For example, if the first returned document was the most relevant (i.e., the corresponding French language version of the query document) this would result in an ROC with area under the curve (AROC) of 1. If the most relevant document came in above the 75th percentile of all documents, this would lead to an AROC of 0.75, and so on. So, we're basically measuring how highly the corresponding French language document ranks in the retrieval results. For a collection of queries we take the simple average of each query's AROC to obtain the average AROC. An average AROC of 1 is best, a value of 0.5 is as good as chance.

In Table 5, we compare retrieval using sparse CCA to regular CCA. For sparse CCA, we use a sparsity parameter that leads to loadings that are approximately 10% of the full dimensionality, i.e., the canonical components are approximately 90% sparse. We note that sparse CCA is able to achieve good retrieval rates, only slightly sacrificing performance compared to regular CCA. This is the key result of this section: we can achieve performance close to regular CCA, by using only about 12% of the number of loadings (i.e., documents) required by regular CCA. This shows that sparse CCA can narrow in on the most informative dimensions exhibited by data and can be used as an effective dimensionality reduction technique.

In summary, Torres et al. (2007a) illustrates that vocabulary selection using sparse CCA significantly improves the retrieval performance of a computer audition system (by effectively removing noisy words), outperforming a random baseline and a human agreement heuristic.

4.2.2 Vocabulary selection for music information retrieval

In this section we provide a short summary of the results in Torres et al. (2007a), which nicely illustrate how sparse CCA can be used to improve the performance of a statistical musical query application, by identifying problematic query words and eliminating them from the model. The application involves a computer audition system (Turnbull et al. 2008) that can annotate songs with semantically meaningful words or *tags* (such as, e.g., *rock* or *mellow*), or retrieve songs from a database, based on a semantic query. This system is based on a joint probabilistic model between words and acoustic signals, learned from a training data set of songs and song tags. "Noisy" words, that are not or only weakly related to the musical content, will decrease the system's performance and waste computational resources. Sparse CCA is employed to prune away those noisy words and improve the system's performance.

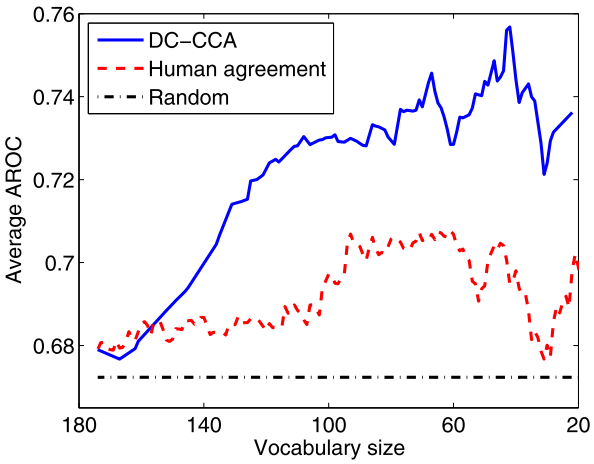

Fig. 5 Comparison of vocabulary selection techniques for music retrieval

The details of this experiment are beyond the scope of this work and can be found in Torres et al. (2007a). In short, each song from the CAL-500 dataset[6] is represented in two different spaces: in a semantic space, based on a bag-of-words representation of a song's semantic tags, and in an audio space, based on Mel-frequency cepstral coefficients (Mckinney 2003) extracted from a song's audio content. This representation allows sparse CCA to identify a small subset of words spanning a semantic subspace that is highly correlated with audio content. In Fig. 5, we use sparse CCA to generate a sequence of vocabularies of progressively smaller size, ranging from full size (containing about 180 words) to very sparse (containing about 20 words), depicted on the horizontal axis. For each vocabulary size, the computer audition system is trained and the average area under the receiver operating characteristic curve (AROC) is shown on the vertical axis, measuring its retrieval performance on an independent test set. The AROC (ranging between 0.5 for randomly ranked retrieval results and 1.0 for a perfect ranking) initially clearly improves, as sparse CCA (DC-CCA) generates vocabularies of smaller size: it is effectively removing noisy words that are detrimental for the system's performance. Also shown in Fig. 5 are the results of training the music retrieval system based on two alternative vocabulary selection techniques: random selection (offering no improvement) and a heuristic that eliminates words exhibiting less agreement amongst the human subjects that were surveyed to collect the CAL-500 dataset (only offering a slight improvement, initially).

4.3 Sparse fisher discriminant analysis

In this section, we show that the FDA problem is an *interesting* special case of the GEV problem and that the special structure of $\boldsymbol{A}$ allows the sparse FDA problem to be solved more efficiently than the general sparse GEV problem.

Let us consider the GEV problem in (GEV-P) with $\boldsymbol{A} \in \mathbb{S}^n_+$, $\boldsymbol{B} \in \mathbb{S}^n_{++}$ and $\operatorname{rank}(\boldsymbol{A}) = 1$. This is exactly the FDA problem as shown in (4) where $\boldsymbol{A}$ is of the form $\boldsymbol{A} = \boldsymbol{a}\boldsymbol{a}^T$, with $\boldsymbol{a} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \in \mathbb{R}^n$. The corresponding GEV problem, written as $\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B}) = \max\{(\boldsymbol{a}^T\boldsymbol{x})^2 : \boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x} = 1\}$, can be posed as a minimization problem as shown by the following result.

[6]The CAL-500 data set consists of a set of songs, annotated with semantic tags, obtained by conducting human surveys. More details can be found in Turnbull et al. (2008).

Proposition 2 *Suppose* $\boldsymbol{x}_1$ *is the solution to*

$$\max_{\boldsymbol{x}} \left\{ (\boldsymbol{a}^T \boldsymbol{x})^2 : \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} = 1 \right\}, \tag{22}$$

and $\boldsymbol{x}_2$ *is the solution to*

$$\min_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} : \boldsymbol{a}^T \boldsymbol{x} = 1 \right\}. \tag{23}$$

Then

$$\boldsymbol{x}_1 = \boldsymbol{x}_2 \sqrt{\boldsymbol{a}^T \boldsymbol{B}^{-1} \boldsymbol{a}}. \tag{24}$$

Proof Consider $\max\{(\boldsymbol{a}^T \boldsymbol{x})^2 : \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} = 1\}$, whose Lagrangian is given as $L_1(\boldsymbol{x}, \lambda_1) = (\boldsymbol{a}^T \boldsymbol{x})^2 - \lambda_1 (\boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} - 1)$, where $\lambda_1 > 0$ is the Lagrange multiplier. Differentiating L_1 w.r.t. $\boldsymbol{x}$ and setting to it zero gives $\boldsymbol{a}\boldsymbol{a}^T \boldsymbol{x} = \lambda_1 \boldsymbol{B} \boldsymbol{x}$, i.e., $\boldsymbol{x} = \lambda_1^{-1} (\boldsymbol{a}^T \boldsymbol{x}) \boldsymbol{B}^{-1} \boldsymbol{a}$. Note that $\boldsymbol{a}^T \boldsymbol{x}$ determines the scale of $\boldsymbol{x}$ and does not determine its direction. Therefore, let $\boldsymbol{x} = c \boldsymbol{B}^{-1} \boldsymbol{a}$, where $c \in \mathbb{R}$ is chosen such that $\boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} = 1$, which gives $\boldsymbol{x}_1 = (\boldsymbol{a}^T \boldsymbol{B}^{-1} \boldsymbol{a})^{-\frac{1}{2}} (\boldsymbol{B}^{-1} \boldsymbol{a})$.

On the other hand, the Lagrangian of $\min\{\boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} : \boldsymbol{a}^T \boldsymbol{x} = 1\}$ is given by $L_2(\boldsymbol{x}, \lambda_2) = \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} - \lambda_2 (\boldsymbol{a}^T \boldsymbol{x} - 1)$, where $\lambda_2 \in \mathbb{R}$ is the Lagrangian multiplier. Differentiating L_2 w.r.t. $\boldsymbol{x}$ and setting it to zero gives $\boldsymbol{x} = \frac{\lambda_2}{2} \boldsymbol{B}^{-1} \boldsymbol{a}$, where λ_2 is chosen by setting $\boldsymbol{a}^T \boldsymbol{x} = 1$, therefore yielding $\boldsymbol{x}_2 = (\boldsymbol{a}^T \boldsymbol{B}^{-1} \boldsymbol{a})^{-1} \boldsymbol{B}^{-1} \boldsymbol{a}$. The result in (24) follows. □

The above result shows that the solutions to (22) and (23) are the same except for scale. Therefore, instead of considering (22), we will consider its equivalent minimization formulation in (23), the advantage of which will become clear when we consider its sparse version, i.e., after introducing the constraint $\{\boldsymbol{x} : \|\boldsymbol{x}\|_0 \leq k\}$ in (23). Clearly, introducing the sparsity constraint in (23) makes it intractable. However, introducing an ℓ_1-norm relaxation in this formulation gives rise to a *convex program*,

$$\min_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} : \boldsymbol{a}^T \boldsymbol{x} = 1,\ \|\boldsymbol{x}\|_1 \leq k \right\}, \tag{25}$$

more specifically a quadratic program (QP). On the other hand, as discussed in Sect. 3, the ℓ_1-norm relaxation to the cardinality constraint, when used in (22), does not convexify the problem (and a convex approximation requires additional relaxation, e.g., as discussed in Appendix A). Therefore, the formulation in (23) is better suited to introduce sparsity than the one in (22).

Suppose that one would like to use a better approximation to $\|\boldsymbol{x}\|_0$ than $\|\boldsymbol{x}\|_1$, for sparse FDA. Using the approximation to $\|\boldsymbol{x}\|_0$ we consider in this work, the sparse version of (23), given by

$$\min_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} + \nu \|\boldsymbol{x}\|_0 : \boldsymbol{a}^T \boldsymbol{x} = 1 \right\}, \tag{26}$$

is approximated as

$$\min_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} + \nu_\varepsilon \sum_{i=1}^{n} \log(\varepsilon + |x_i|) : \boldsymbol{a}^T \boldsymbol{x} = 1 \right\}, \tag{27}$$

where (26) is the regularized version of $\min\{\boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} : \boldsymbol{a}^T \boldsymbol{x} = 1,\ \|\boldsymbol{x}\|_0 \leq k\}$, $\nu > 0$ is the regularization parameter and $\nu_\varepsilon := \nu / \log(1 + \varepsilon^{-1})$. Note that (27) can be written as a d.c. program and therefore, applying the MM method (see Example 1) to (27) results in the following iterative scheme,

$$\boldsymbol{x}^{(l+1)} = \arg\min_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} + \nu_\varepsilon \sum_{i=1}^{n} \frac{|x_i|}{|x_i^{(l)}| + \varepsilon} \,:\, \boldsymbol{a}^T \boldsymbol{x} = 1 \right\}, \tag{28}$$

which is a sequence of QPs.[7] In this case, Algorithm 1 would solve a sequence of QPs as well (since $\boldsymbol{A} \in \mathbb{S}_+^n$, we can have $\tau = 0$).

Suykens et al. (2002, Sect. 3.3) and Mika et al. (2001, Proposition 1) have shown connections between the FDA formulation in (23) with $\boldsymbol{a} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ and $\boldsymbol{B} = \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2$ (see paragraph below (4) for details) and least squares SVMs[8] (classifiers that minimize the squared loss, see Chap. 3 in Suykens et al. 2002). Therefore, sparse FDA is equivalent to feature selection with least squares SVMs, i.e., (25) is equivalent to LASSO, while the formulation in (27) is similar to the one considered in Weston et al. (2003). Since these are well studied problems, we do not pursue further showing the numerical performance of sparse FDA.

5 Conclusion and discussion

This work presents a general, efficient algorithm that allows to obtain sparse solutions to a generalized eigenvalue problem. After proposing a non-convex but tight approximation to the cardinality constraint, we formulate the resulting optimization problem as a d.c. program and derive an iterative algorithm, based on the majorization-minimization method. This results in solving a sequence of quadratically constrained quadratic programs, an algorithm which exhibits global convergence behavior, as we show. To empirically demonstrate the merit of this general framework, specific versions of the proposed algorithm are derived for some special cases, like, e.g., sparse PCA (DC-PCA) and sparse CCA (DC-CCA). For the special case of sparse PCA, we experimentally demonstrate on both benchmark and real-life datasets of varying dimensionality that DC-PCA has performance and scalability similar to the state-of-the-art for sparse PCA, while being derived from a more general framework that can readily be extended to other sparse generalized eigenvalue problems, unlike other sparse PCA algorithms. For the setting of sparse CCA, we illustrate the benefits of DC-CCA in two applications: cross-language document retrieval and vocabulary selection for music information retrieval.

The proposed algorithm does not allow to set the regularization parameter a priori, to guarantee a given sparsity level. SDP-based relaxation methods (see Appendix A), on the other hand, are better suited to achieve a given sparsity level in one shot, by incorporating an explicit constraint on the sparsity of the solution (although, eventually, through relaxation, an approximation of the original problem is solved). Since the algorithm we propose solves a LASSO problem in each step but with a quadratic constraint, in future work, we will explore a modified version of path following techniques like least angle regression (Efron et al. 2004) to learn the entire regularization path.

Another topic for future work is to study the behavior of the solution obtained for (SGEV-A), compared to that of (SGEV-R) as $\varepsilon \to 0$. As mentioned in Sect. 3.1, at present, we do not have any theoretical guarantees about the behavior of $\boldsymbol{x}_\varepsilon$ or $Q_\varepsilon(\boldsymbol{x}_\varepsilon)$ as $\varepsilon \to 0$. In

[7] Although (28) is not a QP in standard form, it can be reformulated as the following standard QP: $\boldsymbol{z}^{(l+1)} = \arg\min_{\boldsymbol{z}} \{\boldsymbol{z}^T \boldsymbol{C}\boldsymbol{z} + \boldsymbol{b}^T \boldsymbol{z} \,:\, \boldsymbol{e}^T \boldsymbol{z} = 1,\ \boldsymbol{F}\boldsymbol{z} \preceq \boldsymbol{0}\}$, where $\boldsymbol{z} := [\boldsymbol{x}^T \ \boldsymbol{y}^T]^T$, $\boldsymbol{C} = \left[\begin{smallmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{smallmatrix}\right]$, $\boldsymbol{b} = \left[\begin{smallmatrix} \boldsymbol{0} \\ \nu_\varepsilon \boldsymbol{G}^{-1}\boldsymbol{1} \end{smallmatrix}\right]$, $\boldsymbol{F} = \left[\begin{smallmatrix} \boldsymbol{I}_n & -\boldsymbol{I}_n \\ -\boldsymbol{I}_n & -\boldsymbol{I}_n \end{smallmatrix}\right]$, $\boldsymbol{G} = \boldsymbol{D}(\boldsymbol{y}^{(l)} + \varepsilon)$ and $\boldsymbol{e} = \left[\begin{smallmatrix} \boldsymbol{a} \\ \boldsymbol{0} \end{smallmatrix}\right]$. Here, y_i are the auxiliary variables introduced for $|x_i|$ (see Boyd and Vandenberghe 2004, Chap. 4). Therefore, for simplicity, we refer to (28) as a QP.

[8] Here, we mean the linear classifier case and not the kernelized version.

future work, we would like to study in detail the conditions under which $Q_\varepsilon(\boldsymbol{x}_\varepsilon) \to Q(\widehat{\boldsymbol{x}})$ as $\varepsilon \to 0$ (see Sect. 3.1 for details on the notation).

Acknowledgements The authors thank the editors and reviewers for their constructive comments. B.K.S. thanks Suvrit Sra for helpful discussions while the former was an intern at the Max Planck Institute for Biological Cybernetics, Tübingen. The authors wish to acknowledge support from the National Science Foundation (grant DMS-MSPA 0625409), the Fair Isaac Corporation and the University of California MICRO program.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

Appendix A: Semidefinite programming relaxation for (SGEV-P)

Let us consider the following approximate program that is obtained by relaxing the non-convex constraint set $\{\boldsymbol{x} : \boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x} = 1\} \cap \{\|\boldsymbol{x}\|_0 \le k\}$ in (SGEV-P):

$$\max_{\boldsymbol{x}} \{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \,:\, \boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x} \le 1,\ \|\boldsymbol{x}\|_1 \le k\}. \tag{29}$$

As mentioned before, this program is still intractable due to the maximization of the non-concave objective. Had the objective function been linear, (29) would have been a canonical convex program, which could then be solved efficiently. One approach to linearize the objective function is by using the *lifting* technique (Lemaréchal and Oustry 1999, Sect. 4.4), which was considered by d'Aspremont et al. (2005) when $\boldsymbol{A} \in \mathbb{S}^n_+$ and $\boldsymbol{B} = \boldsymbol{I}_n$. The lifted version of (29) is given by,

$$\max_{\boldsymbol{X},\boldsymbol{x}} \{\mathrm{tr}(\boldsymbol{X}\boldsymbol{A}) \,:\, \mathrm{tr}(\boldsymbol{X}\boldsymbol{B}) \le 1,\ \|\boldsymbol{x}\|_1 \le k,\ \boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T\}.$$

Note that in the above program, the objective function is linear in $\boldsymbol{X}$, and the constraints are convex except for the non-convex constraint, $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$.[9] Relaxing $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$ to $\boldsymbol{X} - \boldsymbol{x}\boldsymbol{x}^T \in \mathbb{S}^n_+$ results in the following program

$$\max_{\boldsymbol{X},\boldsymbol{x}} \{\mathrm{tr}(\boldsymbol{X}\boldsymbol{A}) \,:\, \mathrm{tr}(\boldsymbol{X}\boldsymbol{B}) \le 1,\ \|\boldsymbol{x}\|_1 \le k,\ \boldsymbol{X} - \boldsymbol{x}\boldsymbol{x}^T \in \mathbb{S}^n_+\}, \tag{30}$$

which is a semidefinite program (SDP). The ℓ_1-norm constraint in (30) can be relaxed as $\|\boldsymbol{x}\|_1^2 \le k^2 \Rightarrow \boldsymbol{1}^T|\boldsymbol{X}|\boldsymbol{1} \le k^2$ so that the problem reduces to solving only for $\boldsymbol{X}$. Here $[|\boldsymbol{X}|]_{ij} = |[\boldsymbol{X}]_{ij}|$. Therefore, we have obtained a tractable convex approximation to (SGEV-P).

Although (30) is a *convex* approximation to (SGEV-P), it is computationally very intensive as general purpose interior-point methods for SDP scale as $O(n^6 \log \epsilon^{-1})$, where ϵ is the required accuracy on the optimal value. For large-scale problems, first-order methods (Nesterov 2005; d'Aspremont et al. 2007) can be used which scale as $O(\epsilon^{-1}n^4\sqrt{\log n})$. Therefore, the SDP-based convex relaxation to (SGEV-P) is prohibitively expensive in computation for large n.

[9] $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T \Leftrightarrow \boldsymbol{X} \in \mathbb{S}^n_+$, $\mathrm{rank}(\boldsymbol{X}) = 1$, where $\mathrm{rank}(\boldsymbol{X}) = 1$ is a non-convex constraint and therefore $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$ is a non-convex constraint.

Appendix B: Approximation to $\|\boldsymbol{x}\|_0$

In Sect. 3.1, we approximated $\|\boldsymbol{x}\|_0$ by

$$\|\boldsymbol{x}\|_\varepsilon := \sum_{i=1}^{n} \frac{\log(1+|x_i|\varepsilon^{-1})}{\log(1+\varepsilon^{-1})}.$$

We now show that $\|\boldsymbol{x}\|_\varepsilon$ is a tighter approximation to $\|\boldsymbol{x}\|_0$ than $\|\boldsymbol{x}\|_1$, for any $\varepsilon > 0$. To this end, let us define $a_\varepsilon := \frac{\log(1+a\varepsilon^{-1})}{\log(1+\varepsilon^{-1})}$, where $a \geq 0$, so that $\|\boldsymbol{x}\|_\varepsilon = \sum_{i=1}^n |x_i|_\varepsilon$. Note that $\|\boldsymbol{x}\|_0 = \lim_{\varepsilon\to 0} \|\boldsymbol{x}\|_\varepsilon$ and

$$\lim_{\varepsilon\to\infty} \|\boldsymbol{x}\|_\varepsilon = \lim_{h\to 0} \sum_{i=1}^{n} \frac{\log(1+|x_i|h)}{\log(1+h)} \overset{(a)}{=} \sum_{i=1}^{n} \lim_{h\to 0} \frac{|x_i|(1+h)}{(1+h|x_i|)} = \sum_{i=1}^{n} |x_i| = \|\boldsymbol{x}\|_1,$$

where we have invoked L'Hôpital's rule in (a). In addition, we have $a > a_{\varepsilon_1} > a_{\varepsilon_2} > \cdots > 1$ for $a > 1$ and $1 > \cdots > a_{\varepsilon_2} > a_{\varepsilon_1} > a$ for $0 < a < 1$, if $\varepsilon_1 > \varepsilon_2 > \cdots$, i.e., for any $a > 0$ and any $0 < \varepsilon < \infty$, the value a_ε is closer to 1 than a is to 1. This means a_ε for any $0 < \varepsilon < \infty$ is a better approximation to $\mathbb{1}_{\{a\neq 0\}}$ than a is to $\mathbb{1}_{\{a\neq 0\}}$. Therefore, $\|\boldsymbol{x}\|_\varepsilon$ for any $0 < \varepsilon < \infty$ is a better approximation to $\|\boldsymbol{x}\|_0$ than $\|\boldsymbol{x}\|_1$ is to $\|\boldsymbol{x}\|_0$.

Appendix C: Derivation of (ALG-R) and (ALG-S)

Let $\boldsymbol{A} \in \mathbb{S}^n_+$ and $\boldsymbol{B} \in \mathbb{S}^n_{++}$. Using $\tau = 0$ in (ALG), we have that $\boldsymbol{x}^{(l+1)}$ is a maximizer of the following program:

$$\begin{aligned}
\max_{\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x}\leq 1} \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}^{(l)} - \frac{\rho_\varepsilon}{2}\left\|\boldsymbol{W}^{(l)}\boldsymbol{x}\right\|_1 &= \max_{\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x}\leq 1} \sum_{i=1}^{n}\left[x_i(\boldsymbol{A}\boldsymbol{x}^{(l)})_i - \frac{\rho_\varepsilon}{2}w_i^{(l)}|x_i|\right] \\
&= \max_{\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x}\leq 1} \sum_{i=1}^{n}|x_i|\left[(\boldsymbol{A}\boldsymbol{x}^{(l)})_i\operatorname{sign}(x_i) - \frac{\rho_\varepsilon}{2}w_i^{(l)}\right] \\
&\overset{(a)}{=} \max_{z^T\boldsymbol{S}\boldsymbol{B}\boldsymbol{S}z\leq 1,\, z\geq\boldsymbol{0}} \sum_{i=1}^{n} z_i\left(\left|(\boldsymbol{A}\boldsymbol{x}^{(l)})_i\right| - \frac{\rho_\varepsilon}{2}w_i^{(l)}\right), \qquad (31)
\end{aligned}$$

where $x_i = s_i z_i$, $z_i \geq 0$ is used in (a) with $s_i := (\boldsymbol{s})_i = \operatorname{sign}((\boldsymbol{A}\boldsymbol{x}^{(l)})_i)$ and $\boldsymbol{S} := \boldsymbol{D}(\boldsymbol{s})$. Note that $\boldsymbol{S}\boldsymbol{B}\boldsymbol{S} \in \mathbb{S}^n_+$. Define $\gamma_i := (\boldsymbol{\gamma})_i = |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i| - \frac{\rho_\varepsilon}{2}w_i^{(l)}$. Then, we have

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{S}\boldsymbol{z}^{(l+1)}, \qquad (32)$$

where $\boldsymbol{z}^{(l+1)}$ is an optimizer of

$$\max_{z}\left\{\boldsymbol{\gamma}^T z : z^T\boldsymbol{S}\boldsymbol{B}\boldsymbol{S}z \leq 1,\, z \succeq \boldsymbol{0}\right\}. \qquad (33)$$

If $\rho_\varepsilon > 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$, then $\boldsymbol{\gamma} \prec \boldsymbol{0}$ and, therefore, $\boldsymbol{z}^{(l+1)} = \boldsymbol{0}$ is the unique optimizer of (33). This implies $\boldsymbol{x}^{(l+1)} = \boldsymbol{0}$ is the unique optimal solution of (ALG). For $\rho_\varepsilon = 2\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$, all the elements of $\boldsymbol{\gamma}$ are negative except for one or more that are zero, i.e., $\gamma_j = 0$ for $j \in \arg\max_i |(\boldsymbol{A}\boldsymbol{x}^{(l)})_i|(w_i^{(l)})^{-1}$ and $\gamma_i < 0$, $i \neq j$. So, we have

$(z^{(l+1)})_i = 0$, $i \neq j$, while any $(z^{(l+1)})_j \geq 0$ is optimal as long as $z^T \boldsymbol{SBS} z \leq 1$. Apart from the uninteresting case $\rho_\varepsilon = 0$ and $\boldsymbol{x}^{(l)} = \mathbf{0}$, having $\rho_\varepsilon = 2\max_i |(\boldsymbol{Ax}^{(l)})_i|(w_i^{(l)})^{-1}$ is clearly rather hypothetical, as ρ_ε is a positive *real* number chosen by the user, and having $\gamma_j = 0$ (and thus many $(z^{(l+1)})_j \geq 0$ optimal) for multiple j even more hypothetical. Therefore, and since $\mathbf{0}$ is a solution of (ALG) when $\rho_\varepsilon = 2\max_i |(\boldsymbol{Ax}^{(l)})_i|(w_i^{(l)})^{-1}$, we set $\boldsymbol{x}^{(l+1)} = \mathbf{0}$ as the optimal solution of (ALG) for $\rho_\varepsilon \geq 2\max_i |(\boldsymbol{Ax}^{(l)})_i|(w_i^{(l)})^{-1}$, when implementing a practical algorithm (e.g., Algorithms 1 and 2).

Let $\rho_\varepsilon < 2\max_i |(\boldsymbol{Ax}^{(l)})_i|(w_i^{(l)})^{-1}$. Consider the Lagrangian of (33):

$$L(z, b, \boldsymbol{\lambda}) = -\boldsymbol{\gamma}^T z + b(z^T \boldsymbol{SBS} z - 1) - \boldsymbol{\lambda}^T z,$$

where $b \geq 0$ and $\boldsymbol{\lambda} \succeq \mathbf{0}$. Differentiating L w.r.t. z and setting it zero gives

$$z^{(l+1)} = \frac{(\boldsymbol{SBS})^+(\boldsymbol{\gamma} + \boldsymbol{\lambda})}{2b}, \tag{34}$$

where $\boldsymbol{C}^+$ represents the Moore-Penrose pseudoinverse of $\boldsymbol{C}$. Substituting for $z^{(l+1)}$ in L gives the following dual program:

$$\min_{b,\boldsymbol{\lambda}} \left\{ b + \frac{1}{4b}(\boldsymbol{\lambda} + \boldsymbol{\gamma})^T (\boldsymbol{SBS})^+ (\boldsymbol{\lambda} + \boldsymbol{\gamma}) : b \geq 0,\ \boldsymbol{\lambda} \succeq \mathbf{0} \right\}. \tag{35}$$

Solving (35) w.r.t. b gives

$$b = \frac{1}{2}\sqrt{(\boldsymbol{\gamma} + \boldsymbol{\lambda})^T (\boldsymbol{SBS})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda})}, \tag{36}$$

which satisfies the constraint, $b \geq 0$, as $(\boldsymbol{SBS})^+ \in \mathbb{S}_+^n$. We show that $b > 0$. For this, let $\mathcal{I} := \{i : s_i \neq 0\}$. Note that for $i \notin \mathcal{I}$, $((\boldsymbol{SBS})^+)_{ij} = 0$, $\forall j$, which means $(\boldsymbol{\gamma} + \boldsymbol{\lambda})^T (\boldsymbol{SBS})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda}) = \sum_{i,j \in \mathcal{I}} (\gamma_i + (\boldsymbol{\lambda})_i)(\gamma_j + (\boldsymbol{\lambda})_j)((\boldsymbol{SBS})^+)_{ij}$. Let $\eta : \mathcal{I} \to \{1, \ldots, |\mathcal{I}|\}$ be a bijective map. By defining $(\bar{\boldsymbol{\gamma}})_{\eta(i)} := (\boldsymbol{\gamma})_i$, $i \in \mathcal{I}$, $(\bar{\boldsymbol{\lambda}})_{\eta(i)} := (\boldsymbol{\lambda})_i$, $i \in \mathcal{I}$, $(\bar{\boldsymbol{s}})_{\eta(i)} = s_i$, $i \in \mathcal{I}$, $\bar{\boldsymbol{S}} = \boldsymbol{D}(\bar{\boldsymbol{s}})$ and $(\bar{\boldsymbol{B}})_{\eta(i),\eta(j)} = \boldsymbol{B}_{ij}$, $i, j \in \mathcal{I}$, we have that $(\boldsymbol{\gamma} + \boldsymbol{\lambda})^T (\boldsymbol{SBS})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda}) = (\bar{\boldsymbol{\gamma}} + \bar{\boldsymbol{\lambda}})^T (\bar{\boldsymbol{S}}\bar{\boldsymbol{B}}\bar{\boldsymbol{S}})^{-1} (\bar{\boldsymbol{\gamma}} + \bar{\boldsymbol{\lambda}}) = (\bar{\boldsymbol{\gamma}} + \bar{\boldsymbol{\lambda}})^T \bar{\boldsymbol{S}} \bar{\boldsymbol{B}}^{-1} \bar{\boldsymbol{S}} (\bar{\boldsymbol{\gamma}} + \bar{\boldsymbol{\lambda}})$. Suppose $b = 0$. This means $\bar{\boldsymbol{\gamma}} + \bar{\boldsymbol{\lambda}} = \mathbf{0}$ as $\bar{\boldsymbol{S}} \bar{\boldsymbol{B}}^{-1} \bar{\boldsymbol{S}} \in \mathbb{S}_{++}^{|\mathcal{I}|}$ (since $\boldsymbol{B} \in \mathbb{S}_{++}^n$ and thus $\bar{\boldsymbol{B}} \in \mathbb{S}_{++}^{|\mathcal{I}|}$), and, therefore, $\bar{\boldsymbol{\lambda}} = -\bar{\boldsymbol{\gamma}}$. However, this violates the constraint $\bar{\boldsymbol{\lambda}} \succeq \mathbf{0}$ because our assumption of $\rho_\varepsilon < 2\max_i |(\boldsymbol{Ax}^{(l)})_i|(w_i^{(l)})^{-1}$ implies that there exists $i \in \{1, \ldots, |\mathcal{I}|\}$ such that $(\bar{\boldsymbol{\gamma}})_i > 0$ (since $(\boldsymbol{\gamma})_j \leq 0$ for $j \notin \mathcal{I}$). Therefore, $b > 0$. Substituting (36) in (35) yields

$$\boldsymbol{\lambda}^{(l+1)} \in \arg\min_{\boldsymbol{\lambda}} \left\{ (\boldsymbol{\gamma} + \boldsymbol{\lambda})^T (\boldsymbol{SBS})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda}) : \boldsymbol{\lambda} \succeq \mathbf{0} \right\}. \tag{37}$$

From the above analysis, it is clear that $\bar{\boldsymbol{\lambda}}^{(l+1)}$ is uniquely computed as

$$\bar{\boldsymbol{\lambda}}^{(l+1)} = \arg\min_{\bar{\boldsymbol{\lambda}}} \left\{ (\bar{\boldsymbol{\gamma}} + \bar{\boldsymbol{\lambda}})^T \bar{\boldsymbol{S}} \bar{\boldsymbol{B}}^{-1} \bar{\boldsymbol{S}} (\bar{\boldsymbol{\gamma}} + \bar{\boldsymbol{\lambda}}) : \bar{\boldsymbol{\lambda}} \succeq \mathbf{0} \right\},$$

because $\bar{\boldsymbol{S}} \bar{\boldsymbol{B}}^{-1} \bar{\boldsymbol{S}} \in \mathbb{S}_{++}^{|\mathcal{I}|}$ and, therefore, the objective is strictly convex in $\bar{\boldsymbol{\lambda}}$, while $(\boldsymbol{\lambda}^{(l+1)})_i$, $i \notin \mathcal{I}$ cannot be uniquely computed. Combining (32), (34), (36) and (37), we have

$$\boldsymbol{x}^{(l+1)} = \frac{\boldsymbol{S}(\boldsymbol{SBS})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})}{\sqrt{(\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})^T (\boldsymbol{SBS})^+ (\boldsymbol{\gamma} + \boldsymbol{\lambda}^{(l+1)})}}. \tag{38}$$

Although $(\boldsymbol{\lambda}^{(l+1)})_i$, $i \notin \mathcal{I}$ cannot be uniquely computed, $(\boldsymbol{x}^{(l+1)})_i$, $i \notin \mathcal{I}$ is uniquely determined as zero because of the pre-multiplication by $\boldsymbol{S}$. Therefore, $\boldsymbol{x}^{(l+1)}$ is the unique optimizer of (31), as shown in (ALG-R).

If $\boldsymbol{B}$ is a diagonal matrix, i.e., $\boldsymbol{B} = \boldsymbol{D}(\boldsymbol{b})$ with $\boldsymbol{b} = (b_1, \ldots, b_n) \succ \mathbf{0}$, then (37) reduces to

$$\boldsymbol{\lambda}^{(l+1)} = \arg\min_{\boldsymbol{\lambda}} \left\{ \sum_{i \in \mathcal{I}} \frac{(\gamma_i + (\boldsymbol{\lambda})_i)^2}{b_i} \,:\, \boldsymbol{\lambda} \succeq \mathbf{0} \right\}.$$

It is easy to see that $(\boldsymbol{\lambda}^{(l+1)})_i = [-\gamma_i]_+$, $\forall\, i \in \mathcal{I}$, where $[a]_+ := \max(0, a)$. Substituting this in (38) yields

$$x_i^{(l+1)} = \frac{s_i [\gamma_i]_+}{b_i \sqrt{\sum_{i=1}^n b_i^{-1} [\gamma_i]_+^2}},$$

as shown in (ALG-S).

Appendix D: Proofs of results in Sect. 3.4

In this section, we present the proofs of Theorem 1 and Corollaries 1–3.

The proof of Theorem 1 is based on a result due to Sriperumbudur and Lanckriet (2009), which is mentioned as Theorem 2 below. In order to understand this result, we introduce some notation and terminology as follows.

The convergence analysis of an iterative procedure like Algorithm 1 uses the notion of a *set-valued mapping*, or *point-to-set mapping*, which is central to the theory of global convergence. A point-to-set map Ψ from a set X into a set Y is defined as $\Psi : X \to \mathscr{P}(Y)$, which assigns a subset of Y with each point of X, where $\mathscr{P}(Y)$ denotes the power set of Y. Ψ is said to be *uniformly compact* on X if there exists a compact set H independent of x such that $\Psi(x) \subset H$ for all $x \in X$. Note that if X is compact, then Ψ is uniformly compact on X. A *fixed point* of the map $\Psi : X \to \mathscr{P}(X)$ is a point x for which $\{x\} = \Psi(x)$.

Many iterative algorithms in mathematical programming can be described using the notion of point-to-set maps. Let X be a set and $x_0 \in X$ a given point. Then an *algorithm*, $\mathcal{A}$, with initial point x_0 is a point-to-set map $\mathcal{A} : X \to \mathscr{P}(X)$ which generates a sequence $\{x_k\}_{k=1}^{\infty}$ via the rule $x_{k+1} \in \mathcal{A}(x_k)$, $k = 0, 1, \ldots$. $\mathcal{A}$ is said to be *globally convergent* if *for any chosen initial point* x_0, the sequence $\{x_k\}_{k=0}^{\infty}$ generated by $x_{k+1} \in \mathcal{A}(x_k)$ (or a subsequence) converges to a point for which a necessary condition of optimality holds: the Karush-Kuhn-Tucker (KKT) conditions in the case of constrained optimization and stationarity in the case of unconstrained optimization.

We now state the convergence result for (14) by Sriperumbudur and Lanckriet (2009), using which we provide the proof of Theorem 1.

Theorem 2 (Sriperumbudur and Lanckriet 2009) *Consider the program,*

$$\min\{u(x) - v(x) \,:\, x \in \Omega\}, \tag{DC}$$

where $\Omega = \{x : c_i(x) \leq 0,\ i \in [m],\ d_j(x) = 0,\ j \in [p]\}$ *and* $[m] := \{1, \ldots, m\}$. *Let* u *and* v *be strictly convex, differentiable functions defined on* $\mathbb{R}^n$. *Also assume* ∇v *is continuous. Let* $\{c_i\}$ *be differentiable convex functions and* $\{d_j\}$ *be affine functions on* $\mathbb{R}^n$. *Suppose* (DC) *is solved iteratively as* $x^{(l+1)} \in \mathcal{A}_{dc}(x^{(l)})$, *where* $\mathcal{A}_{dc}$ *is the point-to-set map defined as*

$$\mathcal{A}_{dc}(y) = \arg\min_{x \in \Omega}\ u(x) - x^T \nabla v(y). \tag{DC-ALG}$$

Let $\{x^{(l)}\}_{l=0}^{\infty}$ be any sequence generated by $\mathcal{A}_{dc}$ defined by (DC-ALG). *Suppose $\mathcal{A}_{dc}$ is uniformly compact on Ω and $\mathcal{A}_{dc}(x)$ is nonempty for any $x \in \Omega$. Then, assuming suitable constraint qualification, all the limit points of $\{x^{(l)}\}_{l=0}^{\infty}$ are fixed points of* (DC-ALG), *which are stationary points of the d.c. program in* (DC), *$u(x^{(l)}) - v(x^{(l)}) \to u(x_*) - v(x_*) =: f^*$ as $l \to \infty$, for some fixed point x_*, $\|x^{(l+1)} - x^{(l)}\| \to 0$, and either $\{x^{(l)}\}_{l=0}^{\infty}$ converges or the set of limit points of $\{x^{(l)}\}_{l=0}^{\infty}$ is a connected and compact subset of $\mathcal{S}(f^*)$, where $\mathcal{S}(a) := \{x \in \mathcal{S} : u(x) - v(x) = a\}$ and $\mathcal{S}$ is the set of fixed points of* (DC-ALG). *If $\mathcal{S}(f^*)$ is finite, then any sequence $\{x^{(l)}\}_{l=0}^{\infty}$ generated by $\mathcal{A}_{dc}$ converges to some x_* in $\mathcal{S}(f^*)$.*

Proof (Theorem 1) Since Algorithm 1 and (ALG) are equivalent, let $\mathcal{A}$ correspond to the point-to-set map in (ALG). As noted before, (ALG) is obtained by applying linear majorization to (9), which is equivalent to (SGEV-A), with $\Omega = \{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \le 1, -\boldsymbol{y} \preceq \boldsymbol{x} \preceq \boldsymbol{y}\}$, $u(\boldsymbol{x}, \boldsymbol{y}) = \tau\|\boldsymbol{x}\|_2^2$ and $v(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T(\boldsymbol{A} + \tau \boldsymbol{I}_n)\boldsymbol{x} - \rho_\varepsilon \sum_{i=1}^n \log(y_i + \varepsilon)$. Let $\tau \neq 0$. It is easy to check that u and v satisfy the conditions of Theorem 2. Define $\boldsymbol{z} := (\boldsymbol{x}^T, \boldsymbol{y}^T)^T$. Note that $\Omega = \{\boldsymbol{z} : \boldsymbol{z}^T \widetilde{\boldsymbol{A}}\boldsymbol{z} \le 1, \widetilde{\boldsymbol{B}}\boldsymbol{z} \preceq \boldsymbol{0}, \widetilde{\boldsymbol{C}}\boldsymbol{z} \succeq \boldsymbol{0}\}$, where $\widetilde{\boldsymbol{A}} = \left[\begin{smallmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{smallmatrix}\right]$, $\widetilde{\boldsymbol{B}} = [\boldsymbol{I}_n \ \ -\boldsymbol{I}_n]$ and $\widetilde{\boldsymbol{C}} = [\boldsymbol{I}_n \ \ \boldsymbol{I}_n]$. $\Omega_1 := \{\boldsymbol{z} : \widetilde{\boldsymbol{B}}\boldsymbol{z} \preceq \boldsymbol{0}\}$ and $\Omega_2 := \{\boldsymbol{z} : \widetilde{\boldsymbol{C}}\boldsymbol{z} \succeq \boldsymbol{0}\}$ are closed sets in $\mathbb{R}^{2n}$, while $\Omega_3 := \{\boldsymbol{z} : \boldsymbol{z}^T \widetilde{\boldsymbol{A}}\boldsymbol{z} \le 1\}$ is compact in $\mathbb{R}^{2n}$. Note that $\Omega = \Omega_1 \cap \Omega_2 \cap \Omega_3$. Since Ω_1 and Ω_2 are closed, $\Omega_1 \cap \Omega_2$ is closed. Since Ω_3 is compact, $(\Omega_1 \cap \Omega_2) \cap \Omega_3$ is also compact, i.e., Ω is compact and therefore $\mathcal{A}$ is uniformly compact. By the Weierstrass theorem[10] (Minoux 1986), it is clear that $\mathcal{A}(\boldsymbol{z})$ is nonempty for any $\boldsymbol{z} \in \Omega$. The result therefore follows from Theorem 2. When $\tau = 0$, all the conditions of Theorem 2 hold except for u being strictly convex. The strict convexity of u in Theorem 2 is mainly needed to ensure that (DC-ALG) has a unique optimizer (for details, see the proof of Theorem 2). Since we showed in Appendix C how Algorithm 1 obtains a unique optimizer in every iteration, when $\tau = 0$, the convergence result also holds for the case of $\tau = 0$. □

Proof (Corollary 1) The proof idea is as follows. First, we consider Algorithm 1 with $\tau = 0$, $\rho_\varepsilon = \rho = 0$ and $\boldsymbol{A} \in \mathbb{S}_+^n$ to show that it reduces to (21). Second, we consider the fixed points of (21) and show that the fixed point of (21), which is a stationary point of (SGEV-A) with $\rho = 0$, is indeed the generalized eigenvector associated with $\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B})$. Therefore, by Theorem 1, the sequence $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ generated by (21) converges to the generalized eigenvector associated with $\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B})$.

Consider Algorithm 1 with $\tau = 0$, $\rho_\varepsilon = \rho = 0$ and $\boldsymbol{A} \in \mathbb{S}_+^n$, which reduces to

$$\boldsymbol{x}^{(l+1)} = \arg\min_{\boldsymbol{x}} \left\{ -\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}^{(l)} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} \le 1 \right\}. \tag{39}$$

Since the objective in (39) is linear in $\boldsymbol{x}$, the minimum occurs at the boundary of the constraint set, i.e., $\{\boldsymbol{x} : \boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} = 1\}$. The corresponding Lagrangian is given by

$$L(\boldsymbol{x}, \mu) = -2\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}^{(l)} + \mu(\boldsymbol{x}^T \boldsymbol{B}\boldsymbol{x} - 1),$$

where $\mu > 0$ is the Lagrange multiplier. Differentiating L w.r.t. $\boldsymbol{x}$ and setting it to zero gives

$$\boldsymbol{x} = \mu^{-1} \boldsymbol{B}^{-1} \boldsymbol{A}\boldsymbol{x}^{(l)},$$

[10] The Weierstrass theorem states: If f is a real continuous function on a compact set $K \subset \mathbb{R}^n$, then the problem $\min\{f(x) : x \in K\}$ has an optimal solution $x^* \in K$.

where solving for μ using $\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x} = 1$ yields (21). So far, we have shown that (39) reduces to (21). Next, let us consider the fixed point, $\boldsymbol{x}^*$ of (21). At $\boldsymbol{x}^*$, (21) reduces to $\boldsymbol{x}^* = ([\boldsymbol{x}^*]^T\boldsymbol{A}\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}^*)^{-\frac{1}{2}}\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}^*$, which can be written as $\boldsymbol{A}\boldsymbol{x}^* = \mu^*\boldsymbol{B}\boldsymbol{x}^*$, where $\mu^* = [\boldsymbol{x}^*]^T\boldsymbol{A}\boldsymbol{x}^*$. Note that $\boldsymbol{x}^*$ is the stationary point of (SGEV-A), when $\rho = 0$. The objective of (39) at $\boldsymbol{x}^*$ is $-[\boldsymbol{x}^*]^T\boldsymbol{A}\boldsymbol{x}^*$, which is minimized when $\boldsymbol{x}^*$ is the generalized eigenvector associated with $\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B})$. Since $\mathscr{S}(-\lambda_{\max}(\boldsymbol{A}, \boldsymbol{B})) = \{\pm\boldsymbol{x}^*\}$, the result follows from Theorem 1. □

Proof (Corollary 2) When $\boldsymbol{B} = \boldsymbol{I}_n$, (21) reduces to $\boldsymbol{x}^{(l+1)} = \frac{\boldsymbol{A}\boldsymbol{x}^{(l)}}{\|\boldsymbol{A}\boldsymbol{x}^{(l)}\|_2}$, which is the power method for computing $\lambda_{\max}(\boldsymbol{A})$. The rest of the result simply follows from Corollary 1. □

Proof (Corollary 3) It is easy to see that when $\boldsymbol{B} = \boldsymbol{I}_n$ and $\rho = 0$, Algorithm 1 reduces to

$$\boldsymbol{x}^{(l+1)} = \arg\min_{\boldsymbol{x}} \left\{\tau\|\boldsymbol{x}\|_2^2 - 2\boldsymbol{x}^T(\boldsymbol{A} + \tau\boldsymbol{I}_n)\boldsymbol{x}^{(l)} : \boldsymbol{x}^T\boldsymbol{x} \le 1\right\}. \tag{40}$$

The Lagrangian of (40) is given by

$$L(\boldsymbol{x}, \mu) = \tau\|\boldsymbol{x}\|_2^2 - 2\boldsymbol{x}^T(\boldsymbol{A} + \tau\boldsymbol{I}_n)\boldsymbol{x}^{(l)} + \mu(\boldsymbol{x}^T\boldsymbol{x} - 1),$$

where $\mu \ge 0$ is the Lagrange multiplier. Differentiating L w.r.t. $\boldsymbol{x}$ and setting it zero gives $\boldsymbol{x} = (\mu + \tau)^{-1}(\boldsymbol{A} + \tau\boldsymbol{I}_n)\boldsymbol{x}^{(l)}$, i.e.,

$$\boldsymbol{x}^{(l+1)} = (\mu^{(l+1)} + \tau)^{-1}(\boldsymbol{A} + \tau\boldsymbol{I}_n)\boldsymbol{x}^{(l)},$$

where

$$\mu^{(l+1)}([\boldsymbol{x}^{(l+1)}]^T\boldsymbol{x}^{(l+1)} - 1) = 0 \quad \text{and} \quad [\boldsymbol{x}^{(l+1)}]^T\boldsymbol{x}^{(l+1)} \le 1.$$

At the fixed point, $\boldsymbol{x}^*$, we have $\boldsymbol{A}\boldsymbol{x}^* = \mu^*\boldsymbol{x}^*$, where $[\boldsymbol{x}^*]^T\boldsymbol{x}^* \le 1$ and $\mu^*([\boldsymbol{x}^*]^T\boldsymbol{x}^* - 1) = 0$. Therefore, $[\boldsymbol{x}^*]^T\boldsymbol{A}\boldsymbol{x}^* = \mu^*$. In addition, the objective of (40) at $\boldsymbol{x}^*$ is given by

$$\psi(\boldsymbol{x}^*) = -2[\boldsymbol{x}^*]^T\boldsymbol{A}\boldsymbol{x}^* - \tau\|\boldsymbol{x}^*\|_2^2 = -2\mu^* - \tau\|\boldsymbol{x}^*\|_2^2,$$

which is minimized when $\mu^* = \lambda_{\max}(\boldsymbol{A})$, i.e., $\boldsymbol{x}^*$ is the eigenvector associated with $\lambda_{\max}(\boldsymbol{A})$. The result therefore follows from Theorem 1 as $\mathscr{S}(-\lambda_{\max}(\boldsymbol{A})) = \{\pm\boldsymbol{x}^*\}$. □

References

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., & Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues. *Cell Biology*, *96*, 6745–6750.

Bioucas-Dias, J., Figueiredo, M., & Oliveira, J. (2006). Total-variation image deconvolution: a majorization-minimization approach. In: *Proc. IEEE international conference on acoustics, speech, and signal processing*, Toulouse, France.

Böhning, D., & Lindsay, B. G. (1988). Monotonicity of quadratic-approximation algorithms. *Annals of the Institute of Statistical Mathematics*, *40*(4), 641–663.

Bonnans, J. F., Gilbert, J. C., Lemaréchal, C., & Sagastizábal, C. A. (2006). *Numerical optimization: theoretical and practical aspects*. Berlin: Springer.

Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Bradley, P. S., & Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. In *Proc. 15th international conf. on machine learning* (pp. 82–90). San Francisco: Kaufmann.

Cadima, J., & Jolliffe, I. (1995). Loadings and correlations in the interpretation of principal components. *Applied Statistics*, *22*, 203–214.

Candes, E. J., Wakin, M., & Boyd, S. (2007). Enhancing sparsity by reweighted ℓ_1 minimization. *The Journal of Fourier Analysis and Applications*, *14*, 877–905.

d'Aspremont, A., El Ghaoui, L., Jordan, M. I., & Lanckriet, G. R. G. (2005). A direct formulation for sparse PCA using semidefinite programming. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems* (Vol. 17, pp. 41–48). Cambridge: MIT Press.

d'Aspremont, A., El Ghaoui, L., Jordan, M. I., & Lanckriet, G. R. G. (2007). A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, *49*(3), 434–448.

d'Aspremont, A., Bach, F. R., & El Ghaoui, L. (2008). Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, *9*, 1269–1294.

Daubechies, I., Defrise, M., & Mol, C. D. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, *57*, 1413–1457.

de Leeuw, J. (1977). Applications of convex analysis to multidimensional scaling. In J. R. Barra, F. Brodeau, G. Romier, & B. V. Cutsem (Eds.), *Recent advantages in statistics* (pp. 133–146). Amsterdam: North Holland.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*, 1–38.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, *32*(2), 407–499.

Fazel, M., Hindi, H., & Boyd, S. (2003). Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices. In: *Proc. American control conference*, Denver, Colorado.

Figueiredo, M. A. T., Bioucas-Dias, J. M., & Nowak, R. D. (2007). Majorization-minimization algorithms for wavelet-based image restoration. *IEEE Transactions on Image Processing*, *16*, 2980–2991.

Germann, U. (2001) *Aligned Hansards of the 36th parliament of Canada*. http://www.isi.edu/natural-language/download/hansard/.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M. K., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., & Lander, E. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, *286*, 531–537.

Heiser, W. J. (1987). Correspondence analysis with least absolute residuals. *Computational Statistics and Data Analysis*, *5*, 337–356.

Horst, R., & Thoai, N. V. (1999). D.c. programming: overview. *Journal of Optimization Theory and Applications*, *103*, 1–43.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*, 417–441.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, *28*, 321–377.

Huber, P. J. (1981). *Robust statistics*. New York: Wiley.

Hunter, D. R., & Lange, K. (2004). A tutorial on MM algorithms. *The American Statistician*, *58*, 30–37.

Hunter, D. R., & Li, R. (2005). Variable selection using MM algorithms. *The Annals of Statistics*, *33*, 1617–1642.

Jeffers, J. (1967). Two case studies in the application of principal components. *Applied Statistics*, *16*, 225–236.

Jolliffe, I. (1986). *Principal component analysis*. New York: Springer.

Jolliffe, I. T., Trendafilov, N. T., & Uddin, M. (2003). A modified principal component technique based on the LASSO. *Journal of Computational and Graphical Statistics*, *12*, 531–547.

Journée, M., Nesterov, Y., Richtárik, P., & Sepulchre, R. (2010). Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, *11*, 517–553.

Lange, K., Hunter, D. R., & Yang, I. (2000). Optimization transfer using surrogate objective functions with discussion. *Journal of Computational and Graphical Statistics*, *9*(1), 1–59.

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems* (Vol. 13, pp. 556–562). Cambridge: MIT Press.

Lemaréchal, C., & Oustry, F. (1999). *Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization* (Tech. Rep. RR3710). INRIA.

Littman, M. L., Dumais, S. T., & Landauer, T. K. (1998). Automatic cross-language information retrieval using latent semantic indexing. In G. Grefenstette (Ed.), *Cross-language information retrieval* (pp. 51–62). Norwell: Kluwer Academic.

Mackey, L. (2009). Deflation methods for sparse PCA. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems* (Vol. 21, pp. 1017–1024). Cambridge: MIT Press.

McCabe, G. (1984). Principal variables. *Technometrics*, *26*, 137–144.

Mckinney, M. F. (2003). Features for audio and music classification. In *Proc. of the international symposium on music information retrieval* (pp. 151–158).
Meng, X. L. (2000). Discussion on "optimization transfer using surrogate objective functions". *Journal of Computational and Graphical Statistics*, *9*(1), 35–43.
Mika, S., Rätsch, G., & Müller, K. R. (2001). A mathematical programming approach to the kernel Fisher algorithm. In T. Leen, T. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems* (Vol. 13). Cambridge: MIT Press.
Minoux, M. (1986). *Mathematical programming: theory and algorithms*. New York: Wiley.
Moghaddam, B., Weiss, Y., & Avidan, S. (2007a). Generalized spectral bounds for sparse LDA. In: *Proc. of international conference on machine learning*.
Moghaddam, B., Weiss, Y., & Avidan, S. (2007b). Spectral bounds for sparse PCA: exact and greedy algorithms. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems* (Vol. 19). Cambridge: MIT Press.
Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming, Series A*, *103*, 127–152.
Ortega, J. M., & Rheinboldt, W. C. (1970). *Iterative solution of nonlinear equations in several variables*. New York: Academic Press.
Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J., Poggio, T., Gerald, W., Loda, M., Lander, E., & Golub, T. (2001). Multiclass cancer diagnosis using tumor gene expression signature. *Proceedings of the National Academy of Sciences*, *98*, 15,149–15,154.
Rockafellar, R. T. (1970). *Convex analysis*. Princeton: Princeton University Press.
Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.
Sriperumbudur, B. K., & Lanckriet, G. R. G. (2009). On the convergence of the concave-convex procedure. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 22, pp. 1759–1767). Cambridge: MIT Press.
Sriperumbudur, B. K., Torres, D. A., & Lanckriet, G. R. G. (2007). Sparse eigen methods by d.c. programming. In: *Proc. of the 24th annual international conference on machine learning*.
Sriperumbudur, B. K., Torres, D. A., & Lanckriet, G. R. G. (2009). A d.c. programming approach to the sparse generalized eigenvalue problem. In: *Optimization for machine learning workshop, NIPS*.
Strang, G. (1986). *Introduction to applied mathematics*. Wellesley: Cambridge Press.
Suykens, J. A. K., Gestel, T. V., Brabanter, J. D., Moor, B. D., & Vandewalle, J. (2002). *Least squares support vector machines*. Singapore: World Scientific.
Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of Royal Statistical Society, Series B*, *58*(1), 267–288.
Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–244.
Torres, D., Turnbull, D., Barrington, L., & Lanckriet, G. R. G. (2007a). Identifying words that are musically meaningful. In: *Proc. of international symposium on music information and retrieval*.
Torres, D. A., Turnbull, D., Sriperumbudur, B. K., Barrington, L., & Lanckriet, G. R. G. (2007b). Finding musically meaningful words using sparse CCA. In: *Music brain & cognition workshop, NIPS*.
Turnbull, D., Barrington, L., Torres, D., & Lanckriet, G. R. G. (2008). Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, *16*, 467–476.
Vandenberghe, L., & Boyd, S. (1996). Semidefinite programming. *SIAM Review*, *38*, 49–95.
Vinokourov, A., Shawe-Taylor, J., & Cristianini, N. (2003). Inferring a semantic representation of text via cross-language correlation analysis. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems* (Vol. 15, pp. 1473–1480). Cambridge: MIT Press.
Weston, J., Elisseeff, A., Schölkopf, B., & Tipping, M. (2003). Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, *3*, 1439–1461.
Yuille, A. L., & Rangarajan, A. (2003). The concave-convex procedure. *Neural Computation*, *15*, 915–936.
Zangwill, W. I. (1969). *Nonlinear programming: a unified approach*. Englewood Cliffs: Prentice-Hall.
Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, *67*, 301–320.
Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, *15*, 265–286.