J Glob Optim (2016) 65:657–676
DOI 10.1007/s10898-016-0402-z

Projection algorithms for nonconvex minimization with application to sparse principal component analysis

William W. Hager[1] · **Dzung T. Phan**[2] · **Jiajie Zhu**[1,3]

Received: 29 March 2015 / Accepted: 11 January 2016 / Published online: 1 February 2016

Abstract We consider concave minimization problems over nonconvex sets. Optimization problems with this structure arise in sparse principal component analysis. We analyze both a gradient projection algorithm and an approximate Newton algorithm where the Hessian approximation is a multiple of the identity. Convergence results are established. In numerical experiments arising in sparse principal component analysis, it is seen that the performance of the gradient projection algorithm is very similar to that of the truncated power method and the generalized power method. In some cases, the approximate Newton algorithm with a Barzilai–Borwein Hessian approximation and a nonmonotone line search can be substantially faster than the other algorithms, and can converge to a better solution.

Keywords Sparse principal component analysis · Gradient projection · Nonconvex minimization · Approximate Newton · Barzilai–Borwein method

The authors gratefully acknowledge support by the National Science Foundation under Grants 1115568 and 1522629 and by the Office of Naval Research under Grants N00014-11-1-0068 and N00014-15-1-2048.

✉ William W. Hager
hager@ufl.edu
http://people.clas.ufl.edu/hager/

Dzung T. Phan
phandu@us.ibm.com

Jiajie Zhu
zhuuv@bc.edu
http://people.clas.ufl.edu/zplusj/

[1] Department of Mathematics, University of Florida, PO Box 118105, Gainesville, FL 32611-8105, USA

[2] IBM T. J. Watson Research Center, Yorktown Heights, NY 20598, USA

[3] Present Address: Computer Science Department, Boston College, Chestnut Hill, MA, USA

1 Introduction

Principal component analysis (PCA) is an extremely popular tool in engineering and statistical analysis. It amounts to computing the singular vectors associated with the largest singular values. In its simplest setting, the rank-one approximation, amounts to solving an optimization problem of the form

$$\max\{x^{\mathsf{T}}\Sigma x : x \in \mathbb{R}^n, \quad \|x\| = 1\}, \tag{1.1}$$

where $\Sigma = A^{\mathsf{T}}A$ is the covariance matrix associated with the data matrix $A \in \mathbb{R}^{m\times n}$ and $\|\cdot\|$ is the Euclidean norm. As pointed out in [27], there is no loss of generality in assuming that Σ is positive definite since

$$x^{\mathsf{T}}\Sigma x + \sigma = x^{\mathsf{T}}(\Sigma + \sigma I)x$$

whenever x is feasible in (1.1).

The lack of interpretability has been a major concern in PCA. Sparse PCA partly addresses this problem by constraining the number of nonzero components of the maximizing x in (1.1). Given a positive integer κ, the sparse PCA problem associated with (1.1) is

$$\max\{x^{\mathsf{T}}\Sigma x : x \in \mathbb{R}^n, \quad \|x\| = 1, \quad \|x\|_0 \le \kappa\}, \tag{1.2}$$

where $\|x\|_0$ denotes the number of nonzero components of x. Due to the sparsity constraint in (1.2), the feasible set is no longer convex, which makes the optimization problem more difficult. Other optimization problems where sparse solutions play an important role include compressed sensing [9,13], basis pursuit [10,32], financial portfolio analysis [31], and model selection [14].

In [8] PCA loadings smaller than a certain tolerance are simply set to zero to produce sparse principal components. More recently, optimization-based approaches have been used to introduce sparsity. For example, in [26] sparsity is achieved using an l_1 relaxation. That is, the problem (1.2) is replaced by

$$\max\{x^{\mathsf{T}}\Sigma x : x \in \mathbb{R}^n, \quad \|x\| = 1, \quad \|x\|_1^2 \le \kappa\}, \tag{1.3}$$

where $\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n|$. The solution of the relaxed problem (1.3) yields an upper bound for the solution of (1.2). In [22] the Rayleigh quotient problem subject to an l_1-constraint is successively maximized using the authors' SCoTLASS algorithm. In [36], the authors formulate a regression problem and propose numerical algorithms to solve it. Their approach can be applied to large-scale data, but it is computationally expensive. In [12] a new semi-definite relaxation is formulated and a greedy algorithm is developed that computes a full set of good solutions for the target number of non-zero coefficients. With total complexity $O(n^3)$, the algorithm is computationally expensive. Other references to algorithms for sparse optimization include [11,18,19,21,25,30].

Our work is largely motivated by [23,27,35]. In [23] both l_1-penalized and l_0-penalized sparse PCA problems are considered and a generalized power method is developed. The numerical experiments show that their approach outperforms earlier algorithms both in solution quality and in computational speed. Recently, [27,35] both consider the l_0-constrained sparse PCA problem and propose an efficient truncated power method. Their algorithms are equivalent and originated from the classic Frank-Wolfe [15] conditional gradient algorithm.

In this paper, we develop both a gradient projection algorithm and an approximate Newton algorithm. Convergence results are established and numerical experiments are given for

sparse PCA problems of the form (1.2). The algorithms have the same $O(\kappa n)$ iteration complexity as the fastest current algorithms, such as those in [23,27,35]. The gradient projection algorithm with unit step size has nearly identical performance as that of the conditional gradient algorithm with unit step size (ConGradU) [27] and the truncated power method (Tpower) [35]. On the other hand, the approximate Newton algorithm can often converge faster to a better objective value than the other algorithms.

The paper is organized as follows. In Sect. 2 we analyze the gradient projection algorithm when the constraint set is nonconvex. Section 3 introduces and analyzes the approximate Newton scheme. Section 4 examines the performance of the algorithms in some numerical experiments based on classic examples found in the sparse PCA literature.

Notation. If $f : R^n \to R$ is differentiable, then $\nabla f(x)$ denotes the gradient of f, a row vector, while $g(x)$ denotes the gradient of f arranged as a column vector. The subscript k denotes the iteration number. In particular, x_k is the k-th x-iterate and $g_k = g(x_k)$. The i-th element of the k-th iterate is denoted x_{ki}. The Euclidean norm is denoted $\|\cdot\|$, while $\|\cdot\|_0$ denotes cardinality (number of non-zero elements). If $x \in \mathbb{R}^n$, then the support of x is the set of indices of nonzeros components:

$$\text{supp}(x) = \{i : x_i \neq 0\}.$$

If $\Omega \subset \mathbb{R}^n$, then $\text{conv}(\Omega)$ is the convex hull of Ω. If $\mathcal{S} \subset \{1, 2, \ldots, n\}$, then $x_{\mathcal{S}}$ is the vector obtained by replacing x_i for $i \in \mathcal{S}^c$ by 0. If $\mathcal{A}$ is a set, then $\mathcal{A}^c$ is its complement.

2 Gradient projection algorithm

Let us consider an optimization problem of the form

$$\min\{f(x) : x \in \Omega\}, \tag{2.1}$$

where $\Omega \subset \mathbb{R}^n$ is a nonempty, closed set and $f : \Omega \to \mathbb{R}$ is differentiable on Ω. Often, the gradient projection algorithm is presented in the context of an optimization problem where the feasible set Ω is convex [3,4,17]. Since the feasible set for the sparse PCA problem (1.2) is nonconvex, we will study the gradient projection algorithm for a potentially nonconvex feasible set Ω.

The projection of x onto Ω is defined by

$$P_\Omega(x) = \arg\min_{y \in \Omega} \|x - y\|.$$

For the constraint set Ω that arises in sparse PCA, the projection can be expressed as follows:

Proposition 2.1 *For the set*

$$\Omega = \{x \in \mathbb{R}^n : \|x\| = 1, \quad \|x\|_0 \leq \kappa\}, \tag{2.2}$$

where κ is a positive integer, we have

$$T_\kappa(x)/\|T_\kappa(x)\| \in P_\Omega(x),$$

where $T_\kappa(x)$ is the vector obtained from x by replacing $n - \kappa$ elements of x with smallest magnitude by 0.

Proof If $y \in \Omega$, then $\|x - y\|^2 = \|x\|^2 + 1 - 2\langle x, y\rangle$. Hence, we have

$$P_\Omega(x) = \arg\ \min\{-\langle x, y\rangle : \|y\| = 1, \quad \|y\|_0 \leq \kappa\}. \tag{2.3}$$

In [27, Prop. 4.3], it is shown that the minimum is attained at $y = T_\kappa(x)/\|T_\kappa(x)\|$. We include the proof since it is short and we need to refer to it later. Given any set $\mathcal{S} \subset \{1, 2, \ldots, n\}$, the solution of the problem

$$\min\{-\langle x, y\rangle : \|y\| = 1, \quad \text{supp}(y) = \mathcal{S}\}$$

is $y = x_\mathcal{S}/\|x_\mathcal{S}\|$ by the Schwarz inequality, and the corresponding objective value is $-\|x_\mathcal{S}\|$, where $x_\mathcal{S}$ is the vector obtained by replacing x_i for $i \in \mathcal{S}^c$ by 0. Clearly, the minimum is attained when $\mathcal{S}$ is the set of indices of x associated with the κ absolute largest components. □

In general, when Ω is closed, the projection exists, although it may not be unique when Ω is nonconvex. If $x_k \in \Omega$ is the current iterate, then in one of the standard implementations of gradient projection algorithm, x_{k+1} is obtained by a line search along the line segment connecting x_k and $P_\Omega(x_k - s_k g_k)$, where g_k is the gradient at x_k and $s_k > 0$ is the step size. When Ω is nonconvex, this line segment is not always contained in Ω. Hence, we will focus on gradient projection algorithms of the form

$$x_{k+1} \in P_\Omega(x_k - s_k g_k). \tag{2.4}$$

Since Ω is closed, x_{k+1} exists for each k. We first observe that $x_{k+1} - x_k$ always forms an obtuse angle with the gradient, which guarantees descent when f is concave.

Lemma 2.2 *If $x_k \in \Omega$, then*

$$\nabla f(x_k)(y - x_k) \le 0 \quad \textit{for all } y \in P_\Omega(x_k - s_k g_k). \tag{2.5}$$

In particular, for $y = x_{k+1}$ this gives

$$\nabla f(x_k)(x_{k+1} - x_k) \le 0. \tag{2.6}$$

If f is concave over conv(Ω)*, then $f(x_{k+1}) \le f(x_k)$.*

Proof If $y \in P_\Omega(x_k - s_k g_k)$, then since $P_\Omega(x_k - s_k g_k)$ is set of elements in Ω closest to $x_k - s_k g_k$, we have

$$\|y - (x_k - s_k g_k)\| \le \|x_k - (x_k - s_k g_k)\| = s_k \|g_k\|. \tag{2.7}$$

By the Schwarz inequality and (2.7), it follows that

$$g_k^\mathsf{T}(y - (x_k - s_k g_k)) \le \|g_k\| \|y - (x_k - s_k g_k)\| \le s_k \|g_k\|^2.$$

We rearrange this inequality to obtain (2.5). If f is concave over conv(Ω), then

$$f(x_{k+1}) \le f(x_k) + \nabla f(x_k)(x_{k+1} - x_k). \tag{2.8}$$

By (2.6), $f(x_{k+1}) \le f(x_k)$. □

The following result is well known.

Proposition 2.3 *If $f : R^n \to R$ is concave and $\Omega \subset \mathbb{R}^n$, then*

$$\inf\{f(x) | x \in \Omega\} = \inf\{f(x) | x \in \text{conv}(\Omega)\}, \tag{2.9}$$

where the first infimum is attained only when the second infimum is attained. Moreover, if f is differentiable at $x^* \in \arg\min\{f(x) : x \in \Omega\}$, then

$$\nabla f(x^*)(y - x^*) \ge 0 \quad \textit{for all } y \in \text{conv}(\Omega). \tag{2.10}$$

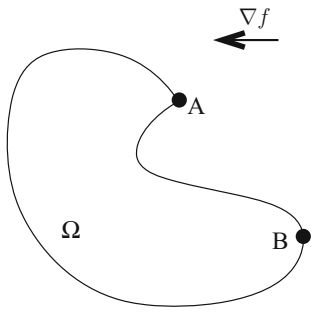

Fig. 1 Example that shows Proposition 2.3 may not hold for local minimizers

Proof The first result (2.9) is proved in [29, Thm. 32.2]. If x^* minimizes $f(x)$ over Ω, then by (2.9),

$$x^* \in \arg\ \min\{f(x) : x \in \mathrm{conv}(\Omega)\}.$$

Since $\mathrm{conv}(\Omega)$ is a convex set, the first-order optimality condition for x^* is (2.10). □

Remark 1 Note that at a local minimizer x^* of f over a nonconvex set Ω, the inequality $\nabla f(x^*)(y - x^*) \geq 0$ may not hold for all $y \in \Omega$. For example, suppose that $f(x) = a^{\mathsf{T}}x$ where $\nabla f = a$ has the direction shown in Fig. 1. The point A is a local minimizer of f over Ω, but (2.10) does not hold. Hence, Proposition 2.3 is only valid for a global minimizer, as stated.

Next, we consider the special choice $y \in P_\Omega(x^* - sg(x^*))$ in Proposition 2.3.

Corollary 2.4 *If $f : R^n \to R$ is concave and*

$$x^* \in \arg\ \min\{f(x) : x \in \Omega\},$$

then

$$\nabla f(x^*)(y - x^*) = 0 \tag{2.11}$$

whenever $y \in P_\Omega(x^* - sg(x^*))$ for some $s \geq 0$.

Proof By Proposition 2.3, we have

$$\nabla f(x^*)(y - x^*) \geq 0$$

for all $y \in P_\Omega(x^* - sg(x^*))$. On the other hand, by Lemma 2.2 with $x_k = x^*$, we have

$$\nabla f(x^*)(y - x^*) \leq 0$$

for all $y \in P_\Omega(x^* - sg(x^*))$. Therefore, (2.11) holds. □

The following property for the projection is needed in the main theorem:

Lemma 2.5 *If Ω is a nonempty closed set, $x_k \in \mathbb{R}^n$ is a sequence converging to x^* and $y_k \in P_\Omega(x_k)$ is a sequence converging to y^*, then $y^* \in P_\Omega(x^*)$.*

Proof Since $y_k \in \Omega$ for each k and Ω is closed, $y^* \in \Omega$. Hence, we have

$$\|y^* - x^*\| \geq \min_{y \in \Omega} \|y - x^*\|.$$

If this inequality is an equality, then we are done; consequently, let us suppose that

$$\|y^* - x^*\| > \min_{y \in \Omega} \|y - x^*\| \geq \min_{y \in \Omega} \{\|y - x_k\| - \|x_k - x^*\|\} = \|y_k - x_k\| - \|x_k - x^*\|.$$

As k tends to ∞, the right side approaches $\|y^* - x^*\|$, which yields a contradiction. □

We now give further justification for the convergence of the gradient projection algorithm in the nonconvex setting.

Theorem 2.6 *If $f : R^n \to R$ is concave, Ω is a compact nonempty set, and x_k is generated by the gradient projection algorithm (2.4), then we have $f(x_{k+1}) \leq f(x_k)$ for each k and*

$$\lim_{k \to \infty} \nabla f(x_k)(x_{k+1} - x_k) = 0. \tag{2.12}$$

If x^* is the limit of any convergent subsequence of the x_k and the step size s_k approaches a limit s^*, then

$$\nabla f(x^*)(y - x^*) \leq 0 \quad \text{for all } y \in P_\Omega(x^* - s^* g(x^*)). \tag{2.13}$$

If f is continuously differentiable around x^*, then

$$\nabla f(x^*)(y^* - x^*) = 0 \tag{2.14}$$

for some $y^* \in P_\Omega(x^* - s^* g(x^*))$.

Proof Sum the concavity inequality (2.8) for $k = 0, 1, \ldots, K-1$ to obtain

$$f(x_K) - f(x_0) \leq \sum_{k=0}^{K-1} \nabla f(x_k)(x_{k+1} - x_k). \tag{2.15}$$

Since f is continuous and Ω is compact, $f^* = \min\{f(x) : x \in \Omega\}$ is finite and

$$f^* - f(x_0) \leq f(x_K) - f(x_0). \tag{2.16}$$

Together, (2.15) and (2.16) yield (2.12) since $\nabla f(x_k)(x_{k+1} - x_k) \leq 0$ for each k by Lemma 2.2.

The relation (2.13) is (2.5) with x_k replaced by x^*. For convenience, let x_k also denote the subsequence of the iterates that converges to x^*, and let $y_k \in P_\Omega(x_k - s_k g_k)$ denote the iterate produced by x_k. Since y_k lies in a compact set, there exists a subsequence converging to a limit y^*. Again, for convenience, let x_k and y_k denote this convergent subsequence. By (2.12) and the fact that f is continuously differentiable around x^*, we have

$$\lim_{k \to \infty} \nabla f(x_k)(y_k - x_k) = \nabla f(x^*)(y^* - x^*) = 0.$$

By Lemma 2.5, $y^* \in P_\Omega(x^* - s^* g(x^*))$. □

Remark 2 The inequalities (2.6), (2.15), and (2.16) imply that

$$\min_{0 \leq k \leq K} \nabla f(x_k)(x_k - x_{k+1}) \leq \frac{f(x_0) - f^*}{K + 1}.$$

When Ω is convex, much stronger convergence results can be established for the gradient projection algorithm. In this case, the projection onto Ω is unique. By [17, Prop. 2.1], for any $x \in \Omega$ and $s > 0$, $x = P_\Omega(x - sg(x))$ if and only if x is a stationary point for (2.1). That is,

$$\nabla f(x)(y - x) \geq 0 \quad \text{for all } y \in \Omega.$$

Moreover, when Ω is convex,

$$\nabla f(x)(P_\Omega(x - sg(x)) - x) \leq -\|P_\Omega(x - \alpha g(x)) - x\|^2/s \tag{2.17}$$

for any $x \in \Omega$ and $s > 0$. Hence, (2.14) implies that the left side of (2.17) vanishes at $x = x^*$, which means that $x^* = P_\Omega(x^* - sg(x^*))$. And conversely, if $x^* = P_\Omega(x^* - sg(x^*))$, then (2.14) holds.

3 Approximate Newton algorithm

To account for second-order information, Bertsekas [4] analyzes the following version of the gradient projection method:

$$x_{k+1} \in P_\Omega(x_k - s_k \nabla^2 f(x_k)^{-1} g_k).$$

Strong convergence results can be established when Ω is convex and f is strongly convex. On the other hand, if f is concave, local minimizers are extreme points of the feasible set, so the analysis is quite different. Suppose that $\nabla^2 f(x_k)$ is approximated by a multiple α_k of the identity matrix as is done in the BB method [2]. This leads to the approximation

$$f(x) \approx f(x_k) + \nabla f(x_k)(x - x_k) + \frac{\alpha_k}{2}\|x - x_k\|^2. \tag{3.1}$$

Let us consider the algorithm in which the new iterate x_{k+1} is obtained by optimizing the quadratic model:

$$x_{k+1} \in \arg\ \min \left\{ \nabla f(x_k)(x - x_k) + \frac{\alpha_k}{2}\|x - x_k\|^2 : x \in \Omega \right\} \tag{3.2}$$

After completing the square, the iteration is equivalent to

$$x_{k+1} \in \arg\ \min \left\{ \alpha_k \|x - (x_k - g_k/\alpha_k)\|^2 : x \in \Omega \right\}.$$

If $\alpha_k > 0$, then this reduces to $x_{k+1} \in P_\Omega(x_k - g_k/\alpha_k)$; in other words, perform the gradient projection algorithm with step size $1/\alpha_k$. If $\alpha_k < 0$, then the iteration reduces to

$$x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k), \tag{3.3}$$

where

$$Q_\Omega(x) = \arg\ \max\{\|x - y\| : y \in \Omega\}. \tag{3.4}$$

If Ω is unbounded, then this iteration does not make sense since the maximum occurs at infinity. But if Ω is compact, then the iteration is justified in the sense that the projection (3.4) exists and the iteration is based on a quadratic model of the function, which could be better than a linear model.

Suppose that $x^* \in Q_\Omega(x^* - g(x^*))/\alpha)$ for some $\alpha < 0$. Due to the equivalence between (3.2) and (3.3), it follows that

$$x^* \in \arg\ \min \left\{ \nabla f(x^*)(x - x^*) + \frac{\alpha}{2}\|x - x^*\|^2 : x \in \Omega \right\}. \tag{3.5}$$

That is, x^* is a global optimizer of the quadratic objective in (3.5) over Ω. Since the objective in the optimization problem (3.5) is concave, Proposition 2.3 yields

$$\nabla f(x^*)(y - x^*) \geq 0 \quad \text{for all } y \in \text{conv}(\Omega). \tag{3.6}$$

Hence, fixed points of the iteration (3.3) satisfy the necessary condition (3.6) associated with a global optimum of (2.1).

In the special case where Ω is the constraint set (2.2) appearing in sparse PCA and $\alpha_k < 0$, the maximization in (3.4) can be evaluated as follows:

Proposition 3.1 *For the set Ω in (2.2) associated with sparse PCA, we have*

$$-T_\kappa(x)/\|T_\kappa(x)\| \in Q_\Omega(x).$$

Proof As in the proof of Proposition 2.1, $\|x - y\|^2 = \|x\|^2 + 1 - 2\langle x, y\rangle$ when y lies in the set Ω of (2.2). Hence, we have

$$Q_\Omega(x) = \arg\ \max\{-\langle x, y\rangle : \|y\| = 1, \quad \|y\|_0 \leq \kappa\}.$$

Given any set $\mathcal{S} \subset \{1, 2, \ldots, n\}$, the solution of the problem

$$\max\{-\langle x, y\rangle : \|y\| = 1, \quad \text{supp}(y) = \mathcal{S}\}$$

is $y = -x_\mathcal{S}/\|x_\mathcal{S}\|$ by the Schwarz inequality, and the corresponding objective value is $\|x_\mathcal{S}\|$. Clearly, the maximum is attained when $\mathcal{S}$ corresponds to a set of indices of x associated with the κ absolute largest components. □

Let us now study the convergence of the iterates generated by the quadratic model (3.2). The case where $\alpha_k > 0$ corresponds to the gradient projection algorithm which was studied in Sect. 2. In this section, we focus on $\alpha_k < 0$ and the iteration $x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k)$. For the numerical experiments, we employ a BB-approximation [2] to the Hessian given by

$$\alpha_k^{\text{BB}} = \frac{(\nabla f(x_k) - \nabla f(x_{k-1}))(x_k - x_{k-1})}{\|x_k - x_{k-1}\|^2}. \tag{3.7}$$

It is well known that the BB-approximation performs much better when it is embedded in a nonmonotone line search. This leads us to study a scheme based on the GLL stepsize rule of [16]. Let $f_k^{\max}$ denote the largest of the M most recent function values:

$$f_k^{\max} = \max\{f(x_{k-j}) : 0 \leq j < \min(k, M)\}.$$

Our convention is that $f_k^{\max} = \infty$ when $M = 0$. The nonmonotone approximate Newton algorithm that we analyze is as follows:

NONMONOTONE APPROXIMATE NEWTON (FOR STRONGLY CONCAVE f)

Given $\sigma \in (0, 1)$, $[\alpha_{\min}, \alpha_{\max}] \subset (-\infty, 0)$, and starting guess x_0.
Set $k = 0$.

Step 1. Choose $\beta_k \in [\alpha_{\min}, \alpha_{\max}]$

Step 2. Set $\alpha_k = \sigma^j \beta_k$ where $j \geq 0$ is the smallest integer such that

$$f(x_{k+1}) \leq f_k^{\max} + (\alpha_k/2)\|x_{k+1} - x_k\|^2 \text{ where}$$

$$x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k)$$

Step 3. If a stopping criterion is satisfied, terminate.

Step 4. Set $k = k + 1$ and go to step 1.

Note that the approximate Newton algorithm is monotone when the memory $M = 1$. If $M = 0$, then the stepsize acceptance condition in Step 2 is satisfied for $j = 0$ and $\alpha_k = \beta_k$. Hence, when $M = 0$, there is no line search. To ensure that β_k lies in the safe-guard interval $[\alpha_{\min}, \alpha_{\max}]$ when using the BB-approximation to the Hessian, it is common to take

$$\beta_k = \text{mid}\,\{\alpha_{\min}, \alpha_k^{BB}, \alpha_{\max}\},$$

where mid denotes median or middle. In the numerical experiments, $\alpha_{\min}$ is large in magnitude and $\alpha_{\max}$ is close to 0, in which case the safe guards have no practical significance; nonetheless, they enter into the convergence theory.

In the approximate Newton algorithm, we make a starting guess β_k for the initial α_k and then increase α_k until the line search acceptance criterion of Step 2 is fulfilled. We first show that for a strongly concave function, the line search criterion is satisfied for j sufficiently large, and the stepsize is uniformly bounded away from zero.

Proposition 3.2 *If f is differentiable on a compact set $\Omega \subset \mathbb{R}^n$ and for some $\mu < 0$, we have*

$$f(y) \le f(x) + \nabla f(x)(y - x) + \frac{\mu}{2}\|y - x\|^2 \quad \textit{for all } x \textit{ and } y \in \Omega, \tag{3.8}$$

then Step 2 in the approximate Newton algorithm terminates with a finite j, and with α_k bounded away from 0, uniformly in k. In particular, we have $\alpha_k \le \bar{\alpha} := \max(\sigma\mu/2, \alpha_{\max}) < 0$.

Proof Since Ω is compact, the set $Q_\Omega(x_k - g_k/\alpha_k)$ is nonempty for each k. If $y \in Q_\Omega(x_k - g_k/\alpha_k)$, then since $Q_\Omega(x_k - s_k g_k)$ is set of elements in Ω farthest from $x_k - g_k/\alpha_k$, we have

$$\|y - (x_k - g_k/\alpha_k)\|^2 \ge \|x_k - (x_k - g_k/\alpha_k)\|^2 = \|g_k\|^2/\alpha_k^2.$$

Rearrange this inequality to obtain

$$\nabla f(x_k)(y - x_k) + \frac{\alpha_k}{2}\|y - x_k\|^2 \le 0 \quad \text{for all } y \in Q_\Omega(x_k - g_k/\alpha_k). \tag{3.9}$$

Substitute $y = x_{k+1}$ and $x = x_k$ in (3.8) and in (3.9), and add to obtain

$$\begin{aligned} f(x_{k+1}) &\le f(x_k) + \left(\frac{\mu - \alpha_k}{2}\right)\|x_{k+1} - x_k\|^2 \\ &\le f_k^{\max} + \left(\frac{\mu - \alpha_k}{2}\right)\|x_{k+1} - x_k\|^2. \end{aligned} \tag{3.10}$$

If $0 > \alpha_k \ge \mu/2$, then $\mu - \alpha_k \le \alpha_k$. Hence, by (3.10), Step 2 must terminate whenever $\alpha_k \ge \mu/2$. Since $\sigma \in (0, 1)$, it follows that Step 2 terminates for a finite j. If Step 2 terminates when $j > 0$, then $\sigma^{j-1}\beta < \mu/2$, which implies that $\alpha_k = \sigma^j\beta < \sigma\mu/2$. If Step 2 terminates when $j = 0$, then $\alpha_k \le \alpha_{\max} < 0$. In either case, $\alpha_k \le \bar{\alpha}$. □

In analyzing the convergence of the approximate Newton algorithm, we also need to consider the continuity of the Q_Ω operator.

Lemma 3.3 *If Ω is a nonempty compact set, $x_k \in \mathbb{R}^n$ is a sequence converging to x^* and $y_k \in Q_\Omega(x_k)$ is a sequence converging to y^*, then $y^* \in Q_\Omega(x^*)$.*

Proof Since $y_k \in \Omega$ for each k and Ω is closed, $y^* \in \Omega$. Hence, we have

$$\|y^* - x^*\| \le \max_{y \in \Omega}\|y - x^*\|.$$

If this inequality is an equality, then we are done; consequently, let us suppose that

$$\|y^* - x^*\| < \max_{y\in\Omega} \|y - x^*\| \leq \max_{y\in\Omega}\{\|y - x_k\| + \|x_k - x^*\|\} = \|y_k - x_k\| + \|x_k - x^*\|.$$

As k tends to ∞, the right side approaches $\|y^* - x^*\|$, which yields a contradiction. □

The following theorem establishes convergence of the approximate Newton algorithm when f is strongly concave.

Theorem 3.4 *If f is continuously differentiable on a compact set $\Omega \subset \mathbb{R}^n$ and (3.8) holds for some $\mu < 0$, then the sequence of objective values $f(x_k)$ generated by the nonmonotone approximate Newton algorithm for memory $M > 0$ converge to a limit f^* as k tends to infinity. If x^* is any accumulation point of the iterates x_k, then $x^* \in Q_\Omega(x^* - g(x^*)/\alpha)$ for some $\alpha < 0$, which implies that (3.5) and (3.6) hold.*

Proof By Proposition 3.2, the stepsize generated by the approximate Newton algorithm satisfies $\alpha_k \leq \bar{\alpha} < 0$. By the line search criterion and the fact that $\alpha_k < 0$, $f(x_{k+1}) \leq f_k^{\max}$, which implies that $f_{k+1}^{\max} \leq f_k^{\max}$ for all k. Since the $f_k^{\max}$ are monotone decreasing and bounded from below, there exists a limit f^*. Since f is continuously differentiable on the compact set Ω, it follows that f is uniformly continuous on Ω. Since the stepsize α_k is uniformly bounded away from zero, the same argument used in the proof of [33, Lem. 4] shows that $f(x_k)$ converges to f^* and $\|x_{k+1} - x_k\|$ tends to zero.

Select any L satisfying $\bar{\alpha} < L < 0$ and let y_k be given by

$$y_k \in \arg\ \min \left\{ \nabla f(x_k)(y - x_k) + \frac{L}{2}\|y - x_k\|^2 : y \in \Omega \right\}. \tag{3.11}$$

It follows that

$$\begin{aligned} \nabla f(x_k)(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|^2 \geq \\ \nabla f(x_k)(y_k - x_k) + \frac{L}{2}\|y_k - x_k\|^2. \end{aligned} \tag{3.12}$$

Since x_{k+1} satisfies (3.2), we have

$$\begin{aligned} \nabla f(x_k)(y_k - x_k) + \frac{\alpha_k}{2}\|y_k - x_k\|^2 \geq \\ \nabla f(x_k)(x_{k+1} - x_k) + \frac{\alpha_k}{2}\|x_{k+1} - x_k\|^2. \end{aligned} \tag{3.13}$$

Add (3.12) and (3.13) to obtain

$$\begin{aligned} \nabla f(x_k)(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|^2 \geq \\ \nabla f(x_k)(x_{k+1} - x_k) + \frac{\alpha_k}{2}\|x_{k+1} - x_k\|^2 + \frac{L - \alpha_k}{2}\|y_k - x_k\|^2. \end{aligned}$$

Since $0 > L > \bar{\alpha} \geq \alpha_k \geq \alpha_{\min} > -\infty$ and $\|x_{k+1} - x_k\|$ approaches zero, we conclude that $\|y_k - x_k\|$ tends to zero. Hence, if x^* is an accumulation point of the iterates x_k, then x^* is an accumulation point of the y_k. Due to the equivalence between (3.2) and (3.3), it follows from (3.11) that $y_k \in Q_\Omega(x_k - g_k/L)$. Since g is continuous, $x^* - g(x^*)/L$ is an accumulation point of $x_k - g_k/L$. By Lemma 3.3, we have $x^* \in Q_\Omega(x^* - g(x^*)/L)$, which completes the proof. □

Due to the equivalence between the inclusion $x^* \in Q_\Omega(x^* - g(x^*))/\alpha)$ and the necessary condition (3.6) for a global optimum, the change $E_k = \|x_{k+1} - x_k\|$ associated with the iteration $x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k)$ could be used to assess convergence of the approximate Newton algorithm. That is, if $x_{k+1} = x_k$, then $x^* = x_k$ satisfies the necessary condition (3.6) for a global optimizer of (2.1). As observed in Theorem 3.4, $E_k = \|x_{k+1} - x_k\|$ tends to zero. We now analyze the convergence rate of E_k.

Theorem 3.5 *If f is continuously differentiable on a compact set $\Omega \subset \mathbb{R}^n$, (3.8) holds for some $\mu < 0$, and the memory $M > 0$, then there exists a constant c, independent of k, such that*

$$\min\{E_j : 0 \le j < kM\} \le \frac{c}{\sqrt{k}}.$$

Proof Let $\ell(k)$ denote the index associated with the M most recent function values:

$$\ell(k) = \arg\ \max\{f(x_j) : \max\{0, k - M\} < j \le k\}.$$

In the approximate Newton algorithm, an acceptable step must satisfy the condition

$$f(x_{k+1}) \le f_k^{\max} + (\alpha_k/2)E_k^2. \tag{3.14}$$

If $k = \ell(j + M) - 1$, then

$$f(x_{k+1}) = f(x_{\ell(j+M)}) = f_{j+M}^{\max}. \tag{3.15}$$

Since $j < \ell(j + M)$, it follows that $j - 1 < \ell(j + M) - 1 = k$, or $j \le k$. During the proof of Theorem 3.4, we observed that the $f_k^{\max}$ sequence is monotone decreasing.

Consequently, $f_k^{\max} \le f_j^{\max}$ since $j \le k$. Use this inequality and (3.15) in (3.14) to obtain

$$f_{j+M}^{\max} \le f_j^{\max} + (\alpha_k/2)E_k^2, \tag{3.16}$$

where $k = \ell(j + M) - 1$. Choose $m > 0$ and sum the inequality (3.16) for $j = 0, M, 2M, \ldots, (m - 1)M$. We have

$$f^* \le f_{mM}^{\max} \le f(x_0) + \sum_{i=1}^{m} (\alpha_{k_i}/2)E_{k_i}^2, \tag{3.17}$$

where $(i - 1)M \le k_i < iM$ and f^* is the limit of the $f_k^{\max}$. Observe that

$$\frac{1}{m}\sum_{i=1}^{m} E_{k_i}^2 \ge \min_{1 \le i \le m} E_{k_i} \ge \min_{0 \le i < mM} E_i^2.$$

Combine this relationship with (3.17) and the bound $\alpha_{k_i} \le \bar{\alpha}$ of Proposition 3.2 to obtain

$$\min_{0 \le i < mM} E_i^2 \le \left(\frac{2(f(x_0) - f^*)}{|\bar{\alpha}|}\right)\frac{1}{m}.$$

Taking the square root of each side, the proof is complete. □

4 Numerical experiments

We will investigate the performance of the gradient project and approximate Newton algorithm relative to previously developed algorithms in the literature. In our experiments we use the gradient projection algorithm with unit stepsize (GPU):

$$x_{k+1} \in P_\Omega(x_k - g_k).$$

And in our experiments with the approximate Newton algorithm, we employ the BB approximation (3.7) for the initial stepsize β_k. The memory is $M = 50$ and $\sigma = 0.25$. This version of the approximate Newton algorithm is denoted GPBB. For the set Ω associated with sparse PCA, we have

$$T_\kappa(x)/\|T_\kappa(x)\| \in P_\Omega(x) \quad \text{and} \quad -T_\kappa(x)/\|T_\kappa(x)\| \in Q_\Omega(x)$$

by Propositions 2.1 and 3.1 respectively.

We compare the performance of our algorithms to those of both the truncated power method (Tpower) [35] and the generalized power method (Gpower) [23]. The conditional gradient algorithm with unit step-size (ConGradU) proposed in [27] is equivalent to the truncated power method. Both truncated and generalized power method are targeted to the sparse PCA problem (1.2). The truncated power method handles the sparsity constraint by pushing the absolute smallest components of the iterates to 0. The iteration can be expressed

$$x_{k+1} = \frac{T_\kappa(-g_k)}{\|T_\kappa(-g_k)\|}. \tag{4.1}$$

For comparison, an iteration of the gradient projection algorithm with unit step size GPU is given by

$$x_{k+1} = \frac{T_\kappa(x_k - g_k)}{\|T_\kappa(x_k - g_k)\|}, \tag{4.2}$$

while the approximate Newton algorithm is

$$x_{k+1} = \operatorname{sgn}(\alpha_k)\frac{T_\kappa(x_k - g_k/\alpha_k^{BB})}{\|T_\kappa(x_k - g_k\alpha_k^{BB})\|}, \tag{4.3}$$

where $\operatorname{sgn}(\alpha) = 1, 0, -1$ depending on whether $\alpha > 0, \alpha = 0$, or $\alpha < 0$ respectively. Since the computation of α_k^{BB} requires the gradient at two points, we start GPBB with one iteration of GPU. For the sparse PCA problem (1.2), the time for one iteration of any of these methods is basically the time to multiply a vector by the covariance matrix Σ. Note that the monotone approximate Newton algorithm could be more costly since the evaluation of an acceptable j may require several evaluations of the objective function.

In the generalized power method, the sparsity constraint is handled using a penalty terms. If $\gamma > 0$ denotes the penalty, then Gpower_{l_1} corresponds to the optimization problem

$$\max_{\|x\|=1} \sqrt{x^\top \Sigma x} - \gamma\|x\|_1,$$

while Gpower_{l_0} corresponds to

$$\max_{\|x\|=1} x^\top \Sigma x - \gamma\|x\|_0.$$

The parameter γ needs to be tuned to achieve the desired cardinality; as γ increases, the cardinality of the Gpower approximation decreases. In contrast, the cardinality is an explicit input

Table 1 Results on Pit props data set

Method	Parameters	Explained variance
GPBB	$\kappa = 6$	0.8939
GPBB	$\kappa = 7$	0.9473
GPU	$\kappa = 6$	0.8939
GPU	$\kappa = 7$	0.9473
Tpower(ConGradU)	$\kappa = 6$	0.8939
Tpower(ConGradU)	$\kappa = 7$	0.9473
Gpower_{l_1}	$\gamma = 0.5(\Leftrightarrow\kappa = 6)$	0.8939
Gpower_{l_1}	$\gamma = 0.4(\Leftrightarrow\kappa = 7)$	0.9473
Gpower_{l_0}	$\gamma = 0.2(\Leftrightarrow\kappa = 6)$	0.8939
Gpower_{l_0}	$\gamma = 0.15(\Leftrightarrow\kappa = 7)$	0.9473

parameter for either the truncated power method or for our algorithms; in many applications, cardinality is often specified.

All experiments were conducted using MATLAB on a GNU/Linux computer with 8GB of RAM and an Intel Core i7-2600 processor. For the starting guess in our experiments, we follow the practice of the Tpower algorithm [35] and set $x = e_i$, the i-th column of the identity matrix, where i is the index of the largest diagonal element of the covariance matrix Σ. Our numerical experiments are based on the sparse PCA problem (1.2). We measure the quality of the solution to (1.2) computed by any of the methods using the ratio $x^T\Sigma x/y^T\Sigma y$ where x is the sparse first principal component computed by any of the algorithms for (1.2) and y is the first principal component (a solution of (1.1)). This ratio is often called the *proportion of the explained variance*.

4.1 Pit props dataset

This dataset [20] contains 180 observations with 13 variables, and a covariance matrix $\Sigma \in R^{13\times 13}$. This is a standard benchmark dataset for Sparse PCA algorithms. We consider $\kappa = 6$ or 7, and we adjust the value of γ to achieve the same sparsity in Gpower. The last column of Table 1 gives the proportion of the explained variance. Observe that all the methods achieve essentially the same proportion of the explained variance.

We also considered multiple sparse principal components for this data set and got the same results as those obtained in Table 2 of [36] for the Tpower and PathSPCA algorithms. Similarly, for the lymphoma data set [1] and the Ramaswamy data set [28], all methods yielded the same proportion of explained variance, although the value of γ for Gpower needed to be tuned to achieve the specified cardinality.

4.2 Randomly generated data

In the next set of experiments, we consider randomly generated data, where $\Sigma = A^TA$ with each entry of $A \in \mathbb{R}^{m\times n}$ generated by a normal distribution with mean 0 and standard deviation 1. For randomly generated matrices, we can study the performance as either m or κ change. Each result that we present is based on an average over 100 randomly generated matrices. In Fig. 2 we plot the proportion of the explained variance versus cardinality for $m = 250$ and $n = 500$. Observe that GPBB yields a significantly better objective value

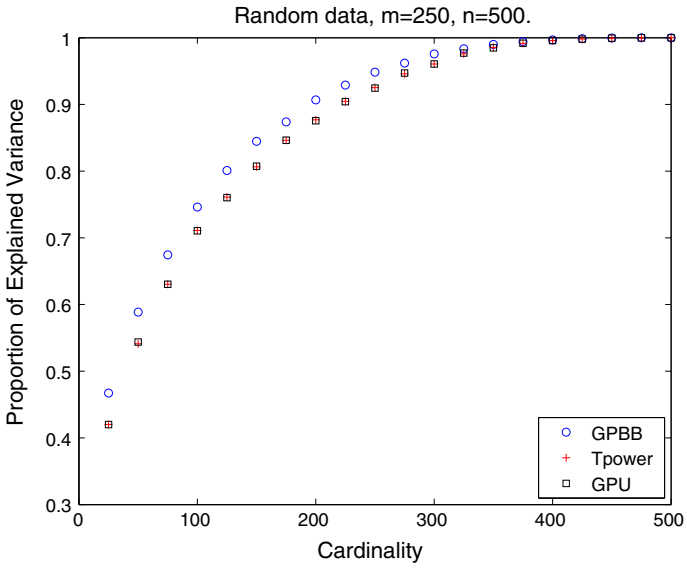

Fig. 2 Explained variance versus cardinality for random data set

as the cardinality decreases when compared to either GPU or ConGradU, while GPU and ConGradU have essentially identical performance. Even though all the algorithms seem to yield similar results in Fig. 2 as the cardinality approaches 500, the convergence of the algorithms is quite different. To illustrate this, let us consider the case where the cardinality is 500. In this case where $\kappa = n$, the sparse PCA problem (1.2) and the original PCA problem (1.1) are identical, and the solution of (1.1) is a normalized eigenvector associated with the largest eigenvalue λ_1 of Σ. Since the optimal objective value is known, we can compute the relative error

$$\frac{\left|\lambda_1^{\text{exact}} - \lambda_1^{\text{approx}}\right|}{\left|\lambda_1^{\text{exact}}\right|},$$

where $\lambda_1^{\text{approx}}$ is the approximation to the optimal objective value generated by any of the algorithms. In Fig. 3 we plot the base 10 logarithm of the relative error versus the iteration number. Observe that GPBB is able to reduce the relative error to the machine precision near 10^{-16} in about 175 iterations, while ConGradU and GPU have relative error around 10^{-3} after 200 iterations. To achieve a relative error around 10^{-16}, ConGradU and GPU require about 4500 iterations, roughly 25 times more than GPBB.

Despite the very nonmonotone behavior of the GPBB iterates, the convergence is relatively fast. The results for the explained variance in Fig. 2 were obtained by running either ConGradU or GPU for 6000 iterations, while GPBB was run for 200 iterations. Hence, the better objective values obtained by GPBB in Fig. 2 were due to the algorithm converging to a better solution, rather than to premature termination of either GPU or ConGradU.

In Fig. 4 we plot the proportion of the explained variance versus the iteration number when $m = 250, n = 500$, and the cardinality $\kappa = 50$ in the random data set. When we plot the function value as in Fig. 4, it is more difficult to see the nonmonotone nature of the convergence for GPBB. This nonmonotone nature is clearly visible in Fig. 3 where we plot

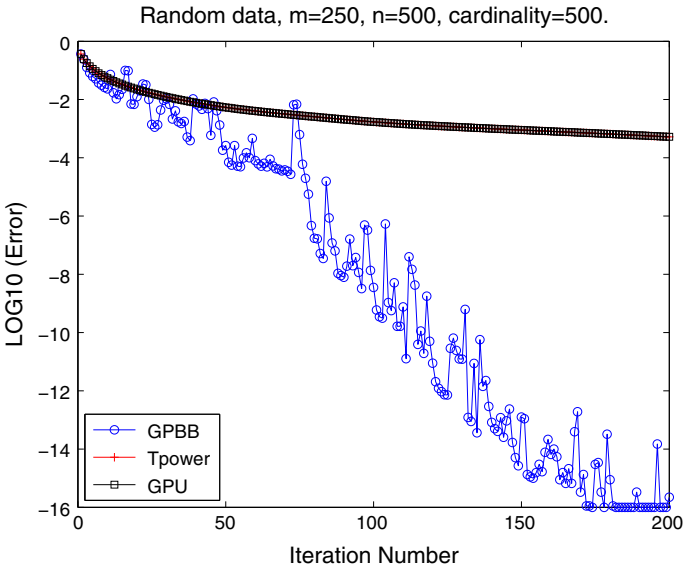

Fig. 3 A plot of the base 10 logarithm of the relative error versus iteration number for the random data set with $m = 250$ and cardinality $\kappa = 500$

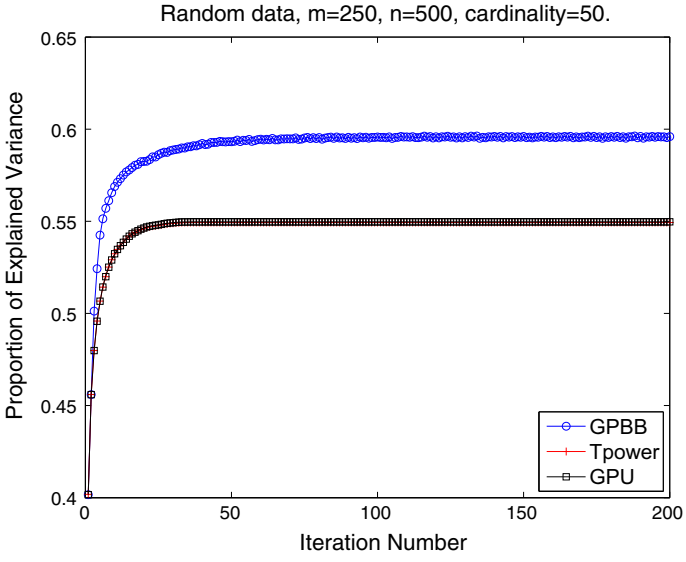

Fig. 4 Explained variance versus iteration number for cardinality 50 in the random data set

the error instead of the function value. In Fig. 5 we show how the explained variance depends on m. As m increases, the explained variance associated with GPBB becomes much better than that of either ConGradU or GPU.

In Fig. 6 we compare the relative error of GPBB for various choices of the memory M. Observe that the monotone scheme where $M = 1$ is slowest, while $M = 50$ gives better performance than that of either a small M or $M = 0$ where $x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k^{BB})$.

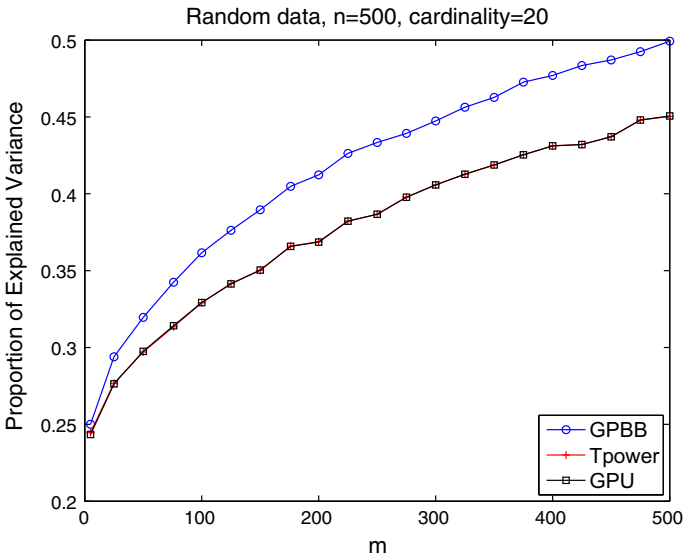

Fig. 5 Explained variance versus m for cardinality 20 in the random data set

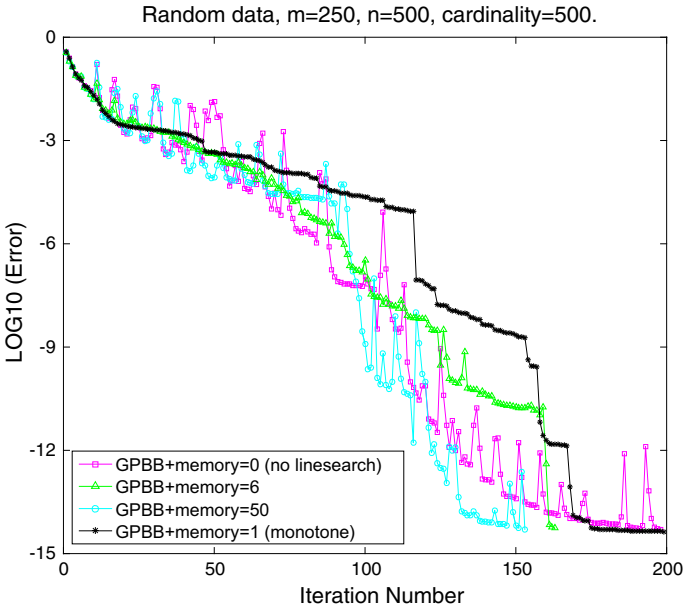

Fig. 6 A plot of the base 10 logarithm of the relative error versus iteration number for the random data set with $m = 250$, cardinality $\kappa = 500$, and various choices for the memory in GPBB

Note that the running time of the monotone scheme was about 4 times larger than that of the nonmonotone schemes since we may need to test several choices of α_k before generating a successful monotone iterate.

To compare with Gpower, we need to choose a value for γ. We first consider a simple case $m = 20$, $n = 20$ and we use the "default" seed in MATLAB to generate this matrix. The

Table 2 Simple random dataset

Method	Cardinality	Explained variance
`GPBB`	$\kappa = 5$	0.8193
`Tpower(ConGradU)`	$\kappa = 5$	0.7913
$\texttt{Gpower}_{l_1}$	$\gamma = 0.18 (\Leftrightarrow \kappa = 5)$	0.7914
$\texttt{Gpower}_{l_0}$	$\gamma = 0.045 (\Leftrightarrow \kappa = 5)$	0.7914

Table 3 Random data set, $m = 250, n = 500$

Method	Cardinality	Explained variance
`GPBB`	$\kappa = 100$	0.7396
`GPBB`	$\kappa = 120$	0.7823
`Tpower(ConGradU)`	$\kappa = 100$	0.7106
`Tpower(ConGradU)`	$\kappa = 120$	0.7536
$\texttt{Gpower}_{l_1}$	$\gamma = 0.075$ (`average` $\kappa = 99$)	0.7288
$\texttt{Gpower}_{l_1}$	$\gamma = 0.0684$ (`average` $\kappa = 120$)	0.7679
$\texttt{Gpower}_{l_0}$	$\gamma = 0.0078$ (`average` $\kappa = 100$)	0.7129
$\texttt{Gpower}_{l_0}$	$\gamma = 0.0066$ (`average` $\kappa = 120$)	0.7557

algorithms are used to extract the first principal component with $\kappa = 5$, and with γ tuned to achieve $\kappa = 5$. The results in Table 2 indicate that Gpower performed similar to Tpower, but not as well as GPBB.

In the next experiment, we consider 100 randomly generated matrices with $m = 250$ and $n = 500$, and with the parameter γ in Gpower chosen to achieve an average cardinality near 100 or 120. As seen in Table 3, $\texttt{Gpower}_{l_0}$ achieves similar values for the proportion of the explained variance as Tpower, while $\texttt{Gpower}_{l_1}$ achieves slightly better results and GPBB achieves the best results.

4.3 Hollywood-2009 dataset, densest *k*-subgraph (DkS)

Given an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V} = \{1, 2, \ldots, n\}$ and edge set $\mathcal{E}$, and given an integer $k \in [1, n]$, the densest k-subgraph (DkS) problem is to find a set of k vertices whose average degree in the subgraph induced by this set is as large as possible. Algorithms for finding DkS are useful tools for analyzing networks. Many techniques have been proposed for solving this problem including [5,24,34]. Mathematically, DkS is equivalent to a binary quadratic programming problem

$$\max\{\pi^{\mathsf{T}} A\pi : \pi \in \mathbb{R}^n, \quad \pi \in \{0, 1\}^n, \quad \|\pi\|_0 = k\}, \tag{4.4}$$

where A is the adjacency matrix of the graph; $a_{ij} = 1$ if $(i, j) \in \mathcal{E}$, while $a_{ij} = 0$ otherwise. We relax the constraints $\pi \in \{0, 1\}^n$ and $\|\pi\|_0 = k$ to $\|\pi\| = \sqrt{k}$ and $\|\pi\|_0 \leq k$, and consider the following relaxed version of (4.4)

Table 4 Hollywood data set

Method	Cardinality	Density $\pi^T A\pi/k$	Ratio $\frac{\pi^T A\pi/k}{\lambda}$
GPBB	$k = 500$	379.40	0.1688
GPBB	$k = 600$	401.22	0.1785
GPBB	$k = 700$	593.24	0.2639
GPBB	$k = 800$	649.67	0.2891
GPBB	$k = 900$	700.38	0.3116
GPBB	$k = 1000$	745.95	0.3319
Tpower(ConGradU)	$k = 500$	190.11	0.0846
Tpower(ConGradU)	$k = 600$	401.21	0.1785
Tpower(ConGradU)	$k = 700$	436.53	0.1942
Tpower(ConGradU)	$k = 800$	649.67	0.2891
Tpower(ConGradU)	$k = 900$	700.44	0.3116
Tpower(ConGradU)	$k = 1000$	745.95	0.3319

$$\max\{\pi^T A\pi : \pi \in \mathbb{R}^n, \quad \|\pi\| = \sqrt{k}, \quad \|\pi\|_0 \le k\}. \tag{4.5}$$

After a suitable scaling of π, this problem reduces to the sparse PCA problem (1.2).

Let us consider the Hollywood-2009 dataset [6,7], which is associated with a graph whose vertices are actors in movies, and an edge joins two vertices whenever the associated actors appear in a movie together. The dataset can be downloaded from the following web site:

http://law.di.unimi.it/datasets.php

The adjacency matrix A is 1139905×1139905. In order to apply Gpower to the relaxed problem, we first factored $A + cI$ into a product of the form $R^T R$ using a Cholesky factorization, where $c > 0$ is taken large enough to make $A + cI$ positive definite. Here, R plays the role of the data matrix. However, one of the steps in the Gpower code updates the data matrix by a rank one matrix, and the rank one matrix caused the updated data matrix to exceed the 200 GB memory on the largest computer readily available for the experiments. Hence, this problem was only solved using Tpower and GPBB. Since the adjacency matrix requires less than 2 GB memory, it easily fit on our 8 GB computer.

In Table 4, we compare the density values $\pi^T A\pi/k$ obtained by the algorithms. In addition, we also computed the largest eigenvalue λ of the adjacency matrix A, and give the ratio of the density to λ. Observe that in 2 of the 6 cases, GPBB obtained a significantly better value for the density when compared to Tpower, while in the other 4 cases, both algorithms converged to the same maximum.

5 Conclusions

The gradient projection algorithm was studied in the case where the constraint set Ω may be nonconvex, as it is in sparse principal component analysis. Each iteration of the gradient projection algorithm satisfied the condition $\nabla f(x_k)(x_{k+1} - x_k) \le 0$. Moreover, if f is concave over $\text{conv}(\Omega)$, then $f(x_{k+1}) \le f(x_k)$ for each k. When a subsequence of the iterates converge to x^*, we obtain in Theorem 2.6 the equality $\nabla f(x^*)(y^* - x^*) = 0$ for

some $y^* \in P_\Omega(x^* - s^* g(x^*))$ where P_Ω projects a point onto Ω and s^* is the limiting step size. When Ω is convex, y^* is unique and the condition $\nabla f(x^*)(y^* - x^*) = 0$ is equivalent to the first-order necessary optimality condition at x^* for a local minimizer.

The approximate Newton algorithm with a positive definite Hessian approximation α_k reduced to the projected gradient algorithm with step size $1/\alpha_k$. On the other hand, when $\alpha_k < 0$, as it is when the objective function is concave and the Hessian approximation is computed by the Barzilai–Borwein formula (3.7), the iteration amounts to taking a step along the positive gradient, and then moving as far away as possible while staying inside the feasible set. In numerical experiments based on sparse principal component analysis, the gradient projection algorithm with unit step size performed similar to both the truncated power method and the generalized power method. On the other hand, in some cases, the approximate Newton algorithm with a Barzilai–Borwein step and a nonmonotone line search could converge much faster to a better objective function value than the other methods.

References

1. Alizadeh, A.A., et al.: Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. Nature **403**, 503–511 (2000)
2. Barzilai, J., Borwein, J.M.: Two point step size gradient methods. IMA J. Numer. Anal. **8**, 141–148 (1988)
3. Beck, A., Teboulle, M.: Mirror descent and nonlinear projected subgradient methods for convex optimization. Oper. Res. Lett. **31**, 167–175 (2003)
4. Bertsekas, D.P.: Projected Newton methods for optimization problems with simple constraints. SIAM J. Control Optim. **20**, 221–246 (1982)
5. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: an $o(n1/4)$ approximation for densest k-subgraph. In: Proceedings of the Forty-Second ACM Symposium on Theory of Computing, ACM, pp. 201–210 (2010)
6. Boldi, P., Rosa, M., Santini, M., Vigna, S.: Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In: Proceedings of the 20th International Conference on World Wide Web, ACM Press (2011)
7. Boldi, P., Vigna, S.: The WebGraph framework I: Compression techniques. In: Proceedings of the Thirteenth International World Wide Web Conference (WWW 2004), Manhattan, USA, ACM Press, pp. 595–601 (2004)
8. Cadima, J., Jolliffe, I.T.: Loading and correlations in the interpretation of principle compenents. J. Appl. Stat. **22**, 203–214 (1995)
9. Candes, E., Wakin, M.: An introduction to compressive sampling. IEEE Signal Process. Mag. **25**, 21–30 (2008)
10. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. **20**, 33–61 (1998)
11. Clarkson, K.L.: Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. ACM Trans. Algorithms (TALG) **6**, 63 (2010)
12. d'Aspremont, A., Bach, F., Ghaoui, L.E.: Optimal solutions for sparse principal component analysis. J. Mach. Learn. Res. **9**, 1269–1294 (2008)
13. Donoho, D.L.: Compressed sensing. IEEE Trans. Inform. Theory **52**, 1289–1306 (2006)
14. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Ann. Stat. **32**, 407–499 (2004)
15. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Nav. Res. Logist. Q **3**, 95–110 (1956)
16. Grippo, L., Lampariello, F., Lucidi, S.: A nonmonotone line search technique for Newton's method. SIAM J. Numer. Anal. **23**, 707–716 (1986)
17. Hager, W.W., Zhang, H.: A new active set algorithm for box constrained optimization. SIAM J. Optim. **17**, 526–557 (2006)
18. Hazan, E., Kale, S.: Projection-free online learning, In: Langford, J., Pineau, J. (eds.) Proceedings of the 29th International Conference on Machine Learning, Omnipress, pp. 521–528 (2012)
19. Jaggi, M.: Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning, vol. 28, pp. 427–435 (2013)
20. Jeffers, J.: Two case studies in the application of principal components. Appl. Stat. **16**, 225–236 (1967)

21. Jenatton, R., Obozinski, G., Bach, F.: Structured sparse principal component analysis. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2010)
22. Jolliffe, I.T., Trendafilov, N.T., Uddin, M.: A modified principal component technique based on the LASSO. J. Comput. Graph. Stat. **12**, 531–547 (2003)
23. Journée, M., Nesterov, Y., Richtárik, P., Sepulchre, R.: Generalized power method for sparse principal component analysis. J. Mach. Learn. Res. **11**, 517–553 (2010)
24. Khuller, S., Saha, B.: On finding dense subgraphs, In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) Automata, Languages and Programming, pp. 597–608. Springer, New York (2009)
25. Lacoste-Julien, S., Jaggi, M., Schmidt, M., Pletscher, P.: Block-coordinate Frank-Wolfe optimization for structural SVMs. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning, vol. 28, pp. 53–61 (2013)
26. Luss, R., Teboulle, M.: Convex approximations to sparse PCA via Lagrangian duality. Oper. Res. Lett. **39**, 57–61 (2011)
27. Luss, R., Teboulle, M.: Conditional gradient algorithms for rank-one matrix approximations with a sparsity constraint. SIAM Rev. **55**, 65–98 (2013)
28. Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C.-H., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J.P., et al.: Multiclass cancer diagnosis using tumor gene expression signatures. Proc. Natl. Acad. Sci. USA **98**, 15149–15154 (2001)
29. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton (1970)
30. Sriperumbudur, B.K., Torres, D.A., Lanckriet, G.R.: A majorization-minimization approach to the sparse generalized eigenvalue problem. Mach. Learn. **85**, 3–39 (2011)
31. Takeda, A., Niranjan, M., Gotoh, J.-Y., Kawahara, Y.: Simultaneous pursuit of out-of-sample performance and sparsity in index tracking portfolios. Comput. Manag. Sci. **10**, 21–49 (2013)
32. van den Berg, E., Friedlander, M.P.: Probing the pareto frontier for basis pursuit solutions. SIAM J. Sci. Comput. **31**, 890–912 (2009)
33. Wright, S.J., Nowak, R.D., Figueiredo, M.A.T.: Sparse reconstruction by separable approximation. IEEE Trans. Signal Process. **57**, 2479–2493 (2009)
34. Ye, Y., Zhang, J.: Approximation of dense-n/2-subgraph and the complement of min-bisection. J. Glob. Optim. **25**, 55–73 (2003)
35. Yuan, X.-T., Zhang, T.: Truncated power method for sparse eigenvalue problems. J. Mach. Learn. Res. **14**, 899–925 (2013)
36. Zou, H., Hastie, T., Tibshirani, R.: Sparse principal component analysis. J. Comput. Graph. Stat. **15**, 265–286 (2006)