Proximal Distance Algorithms: Theory and Practice

Kevin L. Keys   KEVIN.KEYS@UCSF.EDU
Department of Medicine
University of California, San Francisco
CA 94158, USA

Hua Zhou   HUAZHOU@UCLA.EDU
Department of Biostatistics
University of California, Los Angeles
CA 90095-1772, USA

Kenneth Lange   KLANGE@UCLA.EDU
Departments of Biomathematics, Human Genetics, and Statistics
University of California, Los Angeles
CA 90095-1766, USA

Abstract

Proximal distance algorithms combine the classical penalty method of constrained minimization with distance majorization. If $f(\boldsymbol{x})$ is the loss function, and C is the constraint set in a constrained minimization problem, then the proximal distance principle mandates minimizing the penalized loss $f(\boldsymbol{x}) + \frac{\rho}{2} \operatorname{dist}(\boldsymbol{x}, C)^2$ and following the solution $\boldsymbol{x}_\rho$ to its limit as ρ tends to ∞. At each iteration the squared Euclidean distance $\operatorname{dist}(\boldsymbol{x}, C)^2$ is majorized by the spherical quadratic $\|\boldsymbol{x} - P_C(\boldsymbol{x}_k)\|^2$, where $P_C(\boldsymbol{x}_k)$ denotes the projection of the current iterate $\boldsymbol{x}_k$ onto C. The minimum of the surrogate function $f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - P_C(\boldsymbol{x}_k)\|^2$ is given by the proximal map $\operatorname{prox}_{\rho^{-1}f}[P_C(\boldsymbol{x}_k)]$. The next iterate $\boldsymbol{x}_{k+1}$ automatically decreases the original penalized loss for fixed ρ. Since many explicit projections and proximal maps are known, it is straightforward to derive and implement novel optimization algorithms in this setting. These algorithms can take hundreds if not thousands of iterations to converge, but the simple nature of each iteration makes proximal distance algorithms competitive with traditional algorithms. For convex problems, proximal distance algorithms reduce to proximal gradient algorithms and therefore enjoy well understood convergence properties. For nonconvex problems, one can attack convergence by invoking Zangwill's theorem. Our numerical examples demonstrate the utility of proximal distance algorithms in various high-dimensional settings, including a) linear programming, b) constrained least squares, c) projection to the closest kinship matrix, d) projection onto a second-order cone constraint, e) calculation of Horn's copositive matrix index, f) linear complementarity programming, and g) sparse principal components analysis. The proximal distance algorithm in each case is competitive or superior in speed to traditional methods such as the interior point method and the alternating direction method of multipliers (ADMM). Source code for the numerical examples can be found at `https://github.com/klkeys/proxdist`.

Keywords: constrained optimization, EM algorithm, majorization, projection, proximal operator

1. Introduction

The solution of constrained optimization problems is part science and part art. As mathematical scientists explore the largely uncharted territory of high-dimensional nonconvex problems, it is imperative to consider new methods. The current paper studies a class of optimization algorithms that combine Courant's penalty method of optimization (Beltrami, 1970; Courant, 1943) with the notion of a proximal operator (Bauschke and Combettes, 2011; Moreau, 1962; Parikh and Boyd, 2013). The classical penalty method turns constrained minimization of a function $f(\boldsymbol{x})$ over a closed set C into unconstrained minimization. The general idea is to seek the minimum point of a penalized version $f(\boldsymbol{x}) + \rho q(\boldsymbol{x})$ of $f(\boldsymbol{x})$, where the penalty $q(\boldsymbol{x})$ is nonnegative and vanishes precisely on C. If one follows the solution vector $\boldsymbol{x}_\rho$ as ρ tends to ∞, then in the limit one recovers the constrained solution. The penalties of choice in the current paper are squared Euclidean distances $\mathrm{dist}(\boldsymbol{x}, C)^2 = \inf_{\boldsymbol{y} \in C} \|\boldsymbol{x} - \boldsymbol{y}\|^2$.

The formula

$$\mathrm{prox}_f(\boldsymbol{y}) \quad = \quad \mathrm{argmin}_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{y}\|^2 \right] \tag{1}$$

defines the proximal map of a function $f(\boldsymbol{x})$. Here $\|\cdot\|$ is again the standard Euclidean norm, and $f(\boldsymbol{x})$ is typically assumed to be closed and convex. Projection onto a closed convex set C is realized by choosing $f(\boldsymbol{x})$ to be the $0/\infty$ indicator $\delta_C(\boldsymbol{x})$ of C. It is possible to drop the convexity assumption if $f(\boldsymbol{x})$ is nonnegative or coercive. In so doing, $\mathrm{prox}_f(\boldsymbol{y})$ may become multi-valued. For example, the minimum distance from a nonconvex set to an exterior point may be attained at multiple boundary points. The point $\boldsymbol{x}$ in the definition (1) can be restricted to a subset S of Euclidean space by replacing $f(\boldsymbol{x})$ by $f(\boldsymbol{x}) + \delta_S(\boldsymbol{x})$, where $\delta_S(\boldsymbol{x})$ is the indicator of S.

One of the virtues of exploiting proximal operators is that they have been thoroughly investigated. For a large number of functions $f(\boldsymbol{x})$, the map $\mathrm{prox}_{cf}(\boldsymbol{y})$ for $c > 0$ is either given by an exact formula or calculable by an efficient algorithm. The known formulas tend to be highly accurate. This is a plus because the classical penalty method suffers from ill conditioning for large values of the penalty constant. Although the penalty method seldom delivers exquisitely accurate solutions, moderate accuracy suffices for many problems.

There are ample precedents in the optimization literature for the proximal distance principle. Proximal gradient algorithms have been employed for many years in many contexts, including projected Landweber, alternating projection onto the intersection of two or more closed convex sets, the alternating-direction method of multipliers (ADMM), and fast iterative shrinkage thresholding algorithms (FISTA) (Beck and Teboulle, 2009; Combettes and Pesquet, 2011; Landweber, 1951). Applications of distance majorization are more recent (Chi et al., 2014; Lange and Keys, 2014; Xu et al., 2017). The overall strategy consists of replacing the distance penalty $\mathrm{dist}(\boldsymbol{x}, C)^2$ by the spherical quadratic $\|\boldsymbol{x} - \boldsymbol{y}_k\|^2$, where $\boldsymbol{y}_k$ is the projection of the kth iterate $\boldsymbol{x}_k$ onto C. To form the next iterate, one then sets

$$\boldsymbol{x}_{k+1} \quad = \quad \mathrm{prox}_{\rho^{-1}f}(\boldsymbol{y}_k) \quad \text{with} \quad \boldsymbol{y}_k \quad = \quad P_C(\boldsymbol{x}_k).$$

The MM (majorization-minimization) principle guarantees that $\boldsymbol{x}_{k+1}$ decreases the penalized loss. We call the combination of Courant's penalty method with distance majorization the *proximal distance principle*. Algorithms constructed according to the principle are *proximal distance algorithms*.

The current paper extends and deepens our previous preliminary treatments of the proximal distance principle. Details of implementation such as Nesterov acceleration matter in performance. We have found that squared distance penalties tend to work better than exact penalties. In the presence of convexity, it is now clear that every proximal distance algorithm reduces to a proximal gradient algorithm. Hence, convergence analysis can appeal to a venerable body of convex theory. This does not imply that the proximal distance algorithm is limited to convex problems. In fact, its most important applications may well be to nonconvex problems. A major focus of this paper is on practical exploration of the proximal distance algorithm.

In addition to reviewing the literature, the current paper presents some fresh ideas. Among the innovations are: a) recasting proximal distance algorithms with convex losses as concave-convex programs, b) providing new perspectives on convergence for both convex and nonconvex proximal distance algorithms, c) demonstrating the virtue of folding constraints into the domain of the loss, and d) treating in detail seven interesting examples. It is noteworthy that some our new convergence theory is pertinent to more general MM algorithms.

It is our sincere hope to enlist other mathematical scientists in expanding and clarifying this promising line of research. The reviewers of the current paper have correctly pointed out that we do not rigorously justify our choices of the penalty constant sequence ρ_k. The recent paper by Li et al. (2017) may be a logical place to start in filling this theoretical gap. They deal with the problem of minimizing $f(\boldsymbol{x})$ subject to $\boldsymbol{Ax} = \boldsymbol{b}$ through the quadratic penalized objective $f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{Ax} - \boldsymbol{b}\|^2$. For the right choices of the penalty sequence ρ_k, their proximal gradient algorithm achieves a $O(k^{-1})$ rate of convergence for $f(\boldsymbol{x})$ strongly convex. As a substitute, we explore the classical problem of determining how accurately the solution $\boldsymbol{y}_\rho$ of the problem $\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \frac{\rho}{2}q(\boldsymbol{x})^2$ approximates the solution $\boldsymbol{y}$ of the constrained problem $\min_{\boldsymbol{x} \in C} f(\boldsymbol{x})$. Polyak (1971) demonstrates that $f(\boldsymbol{y}) - f(\boldsymbol{y}_\rho) = O(\rho^{-1})$ for a penalty function $q(\boldsymbol{x})$ that vanishes precisely on C. Polyak's proof relies on strong differentiability assumptions. Our proof for the case $q(\boldsymbol{x}) = \text{dist}(\boldsymbol{x}, C)$ relies on convexity and is much simpler.

As a preview, let us outline the remainder of our paper. Section 2 briefly sketches the underlying MM principle. We then show how to construct proximal distance algorithms from the MM principle and distance majorization. The section concludes with the derivation of a few broad categories proximal distance algorithms. Section 3 covers convergence theory for convex problems, while Section 4 provides a more general treatment of convergence for nonconvex problems. To avoid breaking the flow of our exposition, all proofs are relegated to the Appendix. Section 5 discusses our numerical experiments on various convex and nonconvex problems. Section 6 closes by indicating some future research directions.

2. Derivation

The derivation of our proximal distance algorithms exploits the majorization-minimization (MM) principle (Hunter and Lange, 2004; Lange, 2010). In minimizing a function $f(\boldsymbol{x})$, the MM principle exploits a surrogate function $g(\boldsymbol{x} \mid \boldsymbol{x}_k)$ that majorizes $f(\boldsymbol{x})$ around the current iterate $\boldsymbol{x}_k$. Majorization mandates both domination $g(\boldsymbol{x} \mid \boldsymbol{x}_k) \geq f(\boldsymbol{x})$ for all feasible $\boldsymbol{x}$ and tangency $g(\boldsymbol{x}_k \mid \boldsymbol{x}_k) = f(\boldsymbol{x}_k)$ at the anchor $\boldsymbol{x}_k$. If $\boldsymbol{x}_{k+1}$ minimizes $g(\boldsymbol{x} \mid \boldsymbol{x}_k)$, then the

descent property $f(\boldsymbol{x}_{k+1}) \le f(\boldsymbol{x}_k)$ follows from the string of inequalities and equalities

$$f(\boldsymbol{x}_{k+1}) \;\le\; g(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k) \;\le\; g(\boldsymbol{x}_k \mid \boldsymbol{x}_k) \;=\; f(\boldsymbol{x}_k).$$

Clever selection of the surrogate $g(\boldsymbol{x} \mid \boldsymbol{x}_{k+1})$ can lead to a simple algorithm with an explicit update that requires little computation per iterate. The number of iterations until convergence of an MM algorithm depends on how tightly $g(\boldsymbol{x} \mid \boldsymbol{x}_k)$ hugs $f(\boldsymbol{x})$. Constraint satisfaction is built into any MM algorithm. If maximization of $f(\boldsymbol{x})$ is desired, then the objective $f(\boldsymbol{x})$ should dominate the surrogate $g(\boldsymbol{x} \mid \boldsymbol{x}_k)$ subject to the tangency condition. The next iterate $\boldsymbol{x}_{k+1}$ is then chosen to maximize $g(\boldsymbol{x} \mid \boldsymbol{x}_k)$. The minorization-maximization version of the MM principle guarantees the ascent property.

The constraint set C over which the loss $f(\boldsymbol{x})$ is minimized can usually be expressed as an intersection $\cap_{i=1}^m C_i$ of closed sets. It is natural to define the penalty

$$q(\boldsymbol{x}) \;=\; \frac{1}{2} \sum_{i=1}^m \alpha_i \operatorname{dist}(\boldsymbol{x}, C_i)^2$$

using a convex combination of the squared distances. The neutral choice $\alpha_i = \frac{1}{m}$ is one we prefer in practice. Distance majorization gives the surrogate function

$$\begin{aligned}
g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) \;&=\; f(\boldsymbol{x}) + \frac{\rho}{2} \sum_{i=1}^m \alpha_i \|\boldsymbol{x} - P_{C_i}(\boldsymbol{x}_k)\|^2 \\
&=\; f(\boldsymbol{x}) + \frac{\rho}{2} \Big\| \boldsymbol{x} - \sum_{i=1}^m \alpha_i P_{C_i}(\boldsymbol{x}_k) \Big\|^2 + c_k
\end{aligned}$$

for an irrelevant constant c_k. If we put $\boldsymbol{y}_k = \sum_{i=1}^m \alpha_i P_{C_i}(\boldsymbol{x}_k)$, then by definition the minimum of the surrogate $g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k)$ occurs at the proximal point

$$\boldsymbol{x}_{k+1} \;=\; \operatorname{prox}_{\rho^{-1} f}(\boldsymbol{y}_k). \tag{2}$$

We call this MM algorithm the proximal distance algorithm. The penalty $q(\boldsymbol{x})$ is generally smooth because

$$\nabla \frac{1}{2} \operatorname{dist}(\boldsymbol{x}, C)^2 \;=\; \boldsymbol{x} - P_C(\boldsymbol{x})$$

at any point $\boldsymbol{x}$ where the projection $P_C(\boldsymbol{x})$ is single valued (Borwein and Lewis, 2006; Lange, 2016). This is always true for convex sets and almost always true for nonconvex sets. For the moment, we will ignore the possibility that $P_C(\boldsymbol{x})$ is multi-valued.

For the special case of projection of an external point $\boldsymbol{z}$ onto the intersection C of the closed sets C_i, one should take $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{z} - \boldsymbol{x}\|^2$. The proximal distance iterates then obey the explicit formula

$$\boldsymbol{x}_{k+1} \;=\; \frac{1}{1 + \rho}(\boldsymbol{z} + \rho \boldsymbol{y}_k).$$

Linear programming with arbitrary convex constraints is another example. Here the loss is $f(\boldsymbol{x}) = \boldsymbol{v}^t \boldsymbol{x}$, and the update reduces to

$$\boldsymbol{x}_{k+1} \;=\; \boldsymbol{y}_k - \frac{1}{\rho}\boldsymbol{v}.$$

If the proximal map is impossible to calculate, but $f(\boldsymbol{x})$ is L-smooth ($\nabla f(\boldsymbol{x})$ is Lipschitz with constant L), then one can substitute the standard majorization

$$f(\boldsymbol{x}) \leq f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^t(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}_k\|^2$$

for $f(\boldsymbol{x})$. Minimizing the sum of the loss majorization plus the penalty majorization leads to the MM update

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \frac{1}{L+\rho}[-\nabla f(\boldsymbol{x}_k) + L\boldsymbol{x}_k + \rho\boldsymbol{y}_k] \\
&= \boldsymbol{x}_k - \frac{1}{L+\rho}[\nabla f(\boldsymbol{x}_k) + \rho\nabla q(\boldsymbol{x}_k)].
\end{aligned} \tag{3}$$

This is a gradient descent algorithm without an intervening proximal map.

In moderate-dimensional problems, local quadratic approximation of $f(\boldsymbol{x})$ can lead to a viable algorithm. For instance, in generalized linear statistical models, Xu et al. (2017) suggest replacing the observed information matrix by the expected information matrix. The latter matrix has the advantage of being positive semidefinite. In our notation, if $\boldsymbol{A}_k \approx d^2 f(\boldsymbol{x}_k)$, then an approximate quadratic surrogate is

$$f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^t(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^t\boldsymbol{A}_k(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{y}_k\|^2.$$

The natural impulse is to update $\boldsymbol{x}$ by the Newton step

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\boldsymbol{A}_k + \rho\boldsymbol{I})^{-1}[\nabla f(\boldsymbol{x}_k) - \rho\boldsymbol{y}_k]. \tag{4}$$

This choice does not necessarily decrease $f(\boldsymbol{x})$. Step halving or another form of backtracking restores the descent property.

A more valid concern is the effort expended in matrix inversion. If $\boldsymbol{A}_k$ is dense and constant, then extracting the spectral decomposition $\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^t$ of $\boldsymbol{A}$ reduces formula (4) to

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{V}(\boldsymbol{D} + \rho\boldsymbol{I})^{-1}\boldsymbol{V}^t[\nabla f(\boldsymbol{x}_k) - \rho\boldsymbol{y}_k],$$

which can be implemented as a sequence of matrix-vector multiplications. Alternatively, one can take just a few terms of the series

$$(\boldsymbol{A}_k + \rho\boldsymbol{I})^{-1} = \rho^{-1}\sum_{j=0}^{\infty}(-\rho^{-1}\boldsymbol{A}_k)^j$$

when ρ is sufficiently large. For a generalized linear model, parameter updating involves solving the linear system

$$(\boldsymbol{Z}^t\boldsymbol{W}_k\boldsymbol{Z} + \rho\boldsymbol{I})\boldsymbol{x} = \boldsymbol{Z}^t\boldsymbol{W}_k^{1/2}\boldsymbol{v}_k + \rho\boldsymbol{y}_k \tag{5}$$

for $\boldsymbol{W}_k$ a diagonal matrix with positive diagonal entries. This task is equivalent to minimizing the least squares criterion

$$\left\|\begin{pmatrix} \boldsymbol{W}_k^{1/2}\boldsymbol{Z} \\ \sqrt{\rho}\boldsymbol{I} \end{pmatrix}\boldsymbol{x} - \begin{pmatrix} \boldsymbol{v}_k \\ \sqrt{\rho}\boldsymbol{y}_k \end{pmatrix}\right\|^2. \tag{6}$$

In the unweighted case, extracting the singular value decomposition $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ facilitates solving the system of equations (5). The svd decomposition is especially cheap if there is a substantial mismatch between the number rows and columns of $\boldsymbol{Z}$. For sparse $\boldsymbol{Z}$, the conjugate gradient algorithm adapted to least squares (Paige and Saunders, 1982b) is subject to much less ill conditioning than the standard conjugate gradient algorithm. Indeed, the algorithm LSQR and its sparse version LSMR (Fong and Saunders, 2011) perform well even when the matrix $(\boldsymbol{Z}^t\boldsymbol{W}_k^{1/2}, \sqrt{\rho}\boldsymbol{I})^t$ is ill conditioned.

The proximal distance principle also applies to unconstrained problems. For example, consider the problem of minimizing a penalized loss $\ell(\boldsymbol{x}) + p(\boldsymbol{A}\boldsymbol{x})$. The presence of the linear transformation $\boldsymbol{A}\boldsymbol{x}$ in the penalty complicates optimization. The strategy of parameter splitting introduces a new variable $\boldsymbol{y}$ and minimizes $\ell(\boldsymbol{x}) + p(\boldsymbol{y})$ subject to the constraint $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. If $P_M(\boldsymbol{z})$ denotes projection onto the manifold

$$M = \{\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}\},$$

then the constrained problem can be solved approximately by minimizing the function

$$\ell(\boldsymbol{x}) + p(\boldsymbol{y}) + \frac{\rho}{2}\operatorname{dist}(\boldsymbol{z}, M)^2$$

for large ρ. If $P_M(\boldsymbol{z}_k)$ consists of two subvectors $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ corresponding to $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$, then the proximal distance updates are

$$\boldsymbol{x}_{k+1} \;\;=\;\; \operatorname{prox}_{\rho^{-1}\ell}(\boldsymbol{u}_k) \quad \text{and} \quad \boldsymbol{y}_{k+1} \;\;=\;\; \operatorname{prox}_{\rho^{-1}p}(\boldsymbol{v}_k).$$

Given the matrix $\boldsymbol{A}$ is $n \times p$, one can attack the projection by minimizing the function

$$q(\boldsymbol{x}) \;\;=\;\; \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{u}\|^2 + \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{v}\|^2.$$

This leads to the solution

$$\boldsymbol{x} \;\;=\;\; (\boldsymbol{I}_p + \boldsymbol{A}^t\boldsymbol{A})^{-1}(\boldsymbol{A}^t\boldsymbol{v} + \boldsymbol{u}) \quad \text{and} \quad \boldsymbol{y} \;\;=\;\; \boldsymbol{A}\boldsymbol{x}.$$

If $n < p$, then the Woodbury formula

$$(\boldsymbol{I}_p + \boldsymbol{A}^t\boldsymbol{A})^{-1} \;\;=\;\; \boldsymbol{I}_p - \boldsymbol{A}^t(\boldsymbol{I}_n + \boldsymbol{A}\boldsymbol{A}^t)^{-1}\boldsymbol{A}$$

reduces the expense of matrix inversion.

Traditionally, convex constraints have been posed as inequalities $C = \{\boldsymbol{x} : a(\boldsymbol{x}) \leq t\}$. Parikh and Boyd (2013) point out how to project onto such sets. The relevant Lagrangian for projecting an external point $\boldsymbol{y}$ amounts to

$$\mathcal{L}(\boldsymbol{x}, \lambda) \;\;=\;\; \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2 + \lambda[a(\boldsymbol{x}) - t]$$

with $\lambda \geq 0$. The corresponding stationarity condition

$$\boldsymbol{0} \;\;=\;\; \boldsymbol{x} - \boldsymbol{y} + \lambda\nabla a(\boldsymbol{x}), \tag{7}$$

can be interpreted as $a[\text{prox}_{\lambda a}(\boldsymbol{y})] = t$. One can solve this one-dimensional equation for λ by bisection. Once λ is available, $\boldsymbol{x} = \text{prox}_{\lambda a}(\boldsymbol{y})$ is available as well. Parikh and Boyd (2013) note that the value $a[\text{prox}_{\lambda a}(\boldsymbol{y})]$ is decreasing in λ. One can verify their claim by implicit differentiation of equation (7). This gives

$$\frac{d}{d\lambda}\boldsymbol{x} = -[\boldsymbol{I} + \lambda d^2 a(\boldsymbol{x})]^{-1}\nabla a(\boldsymbol{x})$$

and consequently the chain rule inequality

$$\frac{d}{d\lambda}a[\text{prox}_{\lambda a}(\boldsymbol{y})] = -da(\boldsymbol{x})[\boldsymbol{I} + \lambda d^2 a(\boldsymbol{x})]^{-1}\nabla a(\boldsymbol{x}) \leq 0.$$

3. Convergence: Convex Case

In the presence of convexity, the proximal distance algorithm reduces to a proximal gradient algorithm. This follows from the representation

$$\boldsymbol{y} = \sum_{i=1}^{m}\alpha_i P_{C_i}(\boldsymbol{x}) = \boldsymbol{x} - \sum_{i=1}^{m}\alpha_i\Big[\boldsymbol{x} - P_{C_i}(\boldsymbol{x})\Big] = \boldsymbol{x} - \nabla q(\boldsymbol{x})$$

involving the penalty $q(\boldsymbol{x})$. Thus, the proximal distance algorithm can be expressed as

$$\boldsymbol{x}_{k+1} = \text{prox}_{\rho^{-1}f}[\boldsymbol{x}_k - \nabla q(\boldsymbol{x}_k)].$$

In this regard, there is the implicit assumption that $q(\boldsymbol{x})$ is 1-smooth. This is indeed the case. According to the Moreau decomposition (Bauschke and Combettes, 2011), for a single closed convex set C

$$\nabla q(\boldsymbol{x}) = \boldsymbol{x} - P_C(\boldsymbol{x}) = \text{prox}_{\delta_C^\star}(\boldsymbol{x}),$$

where $\delta_C^\star(\boldsymbol{x})$ is the Fenchel conjugate of the indicator function

$$\delta_C(\boldsymbol{x}) = \begin{cases} 0 & \boldsymbol{x} \in C \\ \infty & \boldsymbol{x} \notin C. \end{cases}$$

Because proximal operators of closed convex functions are nonexpansive (Bauschke and Combettes, 2011), the result follows for a single set. For the general penalty $q(\boldsymbol{x})$ with m sets, the Lipschitz constants are scaled by the convex coefficients α_i and added to produce an overall Lipschitz constant of 1.

It is enlightening to view the proximal distance algorithm through the lens of concave-convex programming. Recall that the function

$$s(\boldsymbol{x}) = \sup_{\boldsymbol{y} \in C}\left[\boldsymbol{y}^t\boldsymbol{x} - \frac{1}{2}\|\boldsymbol{y}\|^2\right] = \frac{1}{2}\|\boldsymbol{x}\|^2 - \frac{1}{2}\text{dist}(\boldsymbol{x}, C)^2 \qquad (8)$$

is closed and convex for any nonempty closed set C. Danskin's theorem (Lange, 2016) justifies the directional derivative expression

$$d_{\boldsymbol{v}}s(\boldsymbol{x}) = \sup_{\boldsymbol{y} \in P_C(\boldsymbol{x})} \boldsymbol{y}^t\boldsymbol{v} = \sup_{\boldsymbol{y} \in \text{conv } P_C(\boldsymbol{x})} \boldsymbol{y}^t\boldsymbol{v}.$$

This equality allows us to identify the subdifferential $\partial s(\boldsymbol{x})$ as the convex hull conv $P_C(\boldsymbol{x})$. For any $\boldsymbol{y} \in \partial s(\boldsymbol{x}_k)$, the supporting hyperplane inequality entails

$$
\begin{aligned}
\frac{1}{2}\operatorname{dist}(\boldsymbol{x},C)^2 &= \frac{1}{2}\|\boldsymbol{x}\|^2 - s(\boldsymbol{x}) \\
&\leq \frac{1}{2}\|\boldsymbol{x}\|^2 - s(\boldsymbol{x}_k) - \boldsymbol{y}^t(\boldsymbol{x}-\boldsymbol{x}_k) \\
&= \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 + d,
\end{aligned}
$$

where d is a constant not depending on $\boldsymbol{x}$. The same majorization can be generated by rearranging the majorization

$$
\frac{1}{2}\operatorname{dist}(\boldsymbol{x},C)^2 \leq \frac{1}{2}\sum_i \beta_i\|\boldsymbol{x}-\boldsymbol{p}_i\|^2
$$

when $\boldsymbol{y}$ is the convex combination $\sum_i \beta_i \boldsymbol{p}_i$ of vectors $\boldsymbol{p}_i$ from $P_C(\boldsymbol{x}_k)$. These facts demonstrate that the proximal distance algorithm minimizing

$$
f(\boldsymbol{x}) + \frac{\rho}{2}\operatorname{dist}(\boldsymbol{x},C)^2 = f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x}\|^2 - \rho s(\boldsymbol{x})
$$

is a special case of concave-convex programming when $f(\boldsymbol{x})$ is convex. It is worth emphasizing that $f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x}\|^2$ is often strongly convex regardless of whether $f(\boldsymbol{x})$ itself is convex. If we replace the penalty $\operatorname{dist}(\boldsymbol{x},C)^2$ by the penalty $\operatorname{dist}(\boldsymbol{Dx},C)^2$ for a matrix $\boldsymbol{D}$, then the function $s(\boldsymbol{Dx})$ is still closed and convex, and minimization of $f(\boldsymbol{x}) + \frac{\rho}{2}\operatorname{dist}(\boldsymbol{Dx},C)^2$ can also be viewed as an exercise in concave-convex programming.

In the presence of convexity, the proximal distance algorithm is guaranteed to converge. Our exposition relies on well-known operator results (Bauschke and Combettes, 2011). Proximal operators in general and projection operators in particular are nonexpansive and averaged. By definition an averaged operator

$$
M(\boldsymbol{x}) = \alpha\boldsymbol{x} + (1-\alpha)N(\boldsymbol{x})
$$

is a convex combination of a nonexpansive operator $N(\boldsymbol{x})$ and the identity operator $\boldsymbol{I}$. The averaged operators on $\mathbb{R}^p$ with $\alpha \in (0,1)$ form a convex set closed under functional composition. Furthermore, $M(\boldsymbol{x})$ and the base operator $N(\boldsymbol{x})$ share their fixed points. The celebrated theorem of Krasnosel'skii (1955) and Mann (1953) says that if an averaged operator $M(\boldsymbol{x}) = \alpha\boldsymbol{x} + (1-\alpha)N(\boldsymbol{x})$ possesses one or more fixed points, then the iteration scheme $\boldsymbol{x}_{k+1} = M(\boldsymbol{x}_k)$ converges to a fixed point.

These results immediately apply to minimization of the penalized loss

$$
h_\rho(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{\rho}{2}\sum_{i=1}^m \alpha_i \operatorname{dist}(\boldsymbol{x},C_i)^2. \tag{9}
$$

Given the choice $\boldsymbol{y}_k = \sum_{i=1}^m \alpha_i P_{C_i}(\boldsymbol{x}_k)$, the algorithm map $\boldsymbol{x}_{k+1} = \operatorname{prox}_{\rho^{-1}f}(\boldsymbol{y}_k)$ is an averaged operator, being the composition of two averaged operators. Hence, the Krasnosel'skii-Mann theorem guarantees convergence to a fixed point if one or more exist. Now $\boldsymbol{z}$ is a fixed point if and only if

$$
h_\rho(\boldsymbol{z}) \leq f(\boldsymbol{x}) + \frac{\rho}{2}\sum_{i=1}^m \alpha_i\|\boldsymbol{x} - P_{C_i}(\boldsymbol{z})\|^2
$$

for all $\boldsymbol{x}$. In the presence of convexity, this is equivalent to the directional derivative inequality

$$0 \;\; \leq \;\; d_{\boldsymbol{v}} f(\boldsymbol{z}) + \rho \sum_{i=1}^{m} \alpha_i [\boldsymbol{z} - P_{C_i}(\boldsymbol{z})]^t \boldsymbol{v} \;\; = \;\; d_{\boldsymbol{v}} h_\rho(\boldsymbol{z})$$

for all $\boldsymbol{v}$, which is in turn equivalent to $\boldsymbol{z}$ minimizing $h_\rho(\boldsymbol{x})$. Hence, if $h_\rho(\boldsymbol{x})$ attains its minimum value, then the proximal distance iterates converge to a minimum point.

Convergence of the overall proximal distance algorithm is tied to the convergence of the classical penalty method (Beltrami, 1970). In our setting, the loss is $f(\boldsymbol{x})$, and the penalty is $q(\boldsymbol{x}) = \frac{1}{2} \sum_{i=1}^{m} \alpha_i \operatorname{dist}(\boldsymbol{x}, C_i)^2$. Assuming the objective $f(\boldsymbol{x}) + \rho q(\boldsymbol{x})$ is coercive for all $\rho \geq 0$, the theory mandates that the solution path $\boldsymbol{x}_\rho$ is bounded and any limit point of the path attains the minimum value of $f(\boldsymbol{x})$ subject to the constraints. Furthermore, if $f(\boldsymbol{x})$ is coercive and possesses a unique minimum point in the constraint set C, then the path $\boldsymbol{x}_\rho$ converges to that point.

Proximal distance algorithms often converge at a painfully slow rate. Following Mairal (2013), one can readily exhibit a precise bound.

Proposition 1 *Suppose C is closed and convex and $f(\boldsymbol{x})$ is convex. If the point $\boldsymbol{z}$ minimizes $h_\rho(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{\rho}{2} \operatorname{dist}(\boldsymbol{x}, C)^2$, then the proximal distance iterates satisfy*

$$0 \;\; \leq \;\; h_\rho(\boldsymbol{x}_{k+1}) - h_\rho(\boldsymbol{z}) \;\; \leq \;\; \frac{\rho}{2(k+1)} \|\boldsymbol{z} - \boldsymbol{x}_0\|^2.$$

The $O(\rho k^{-1})$ convergence rate of the proximal distance algorithm suggests that one should slowly send ρ to ∞ and refuse to wait until convergence occurs for any given ρ. It also suggests that Nesterov acceleration may vastly improve the chances for convergence. Nesterov acceleration for the general proximal gradient algorithm with loss $\ell(\boldsymbol{x})$ and penalty $p(\boldsymbol{x})$ takes the form

$$\boldsymbol{z}_k \;\; = \;\; \boldsymbol{x}_k + \frac{k-1}{k+d-1}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$$
$$\boldsymbol{x}_{k+1} \;\; = \;\; \operatorname{prox}_{L^{-1}\ell}[\boldsymbol{z}_k - L^{-1} \nabla p(\boldsymbol{z}_k)], \tag{10}$$

where L is the Lipschitz constant for $\nabla p(\boldsymbol{x})$ and d is typically chosen to be 3. Nesterov acceleration achieves an $O(k^{-2})$ convergence rate (Su et al., 2014), which is vastly superior to the $O(k^{-1})$ rate achieved by proximal gradient descent. The Nesterov update possesses the further desirable property of preserving affine constraints. In other words, if $\boldsymbol{A}\boldsymbol{x}_{k-1} = \boldsymbol{b}$ and $\boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{b}$, then $\boldsymbol{A}\boldsymbol{z}_k = \boldsymbol{b}$ as well. In subsequent examples, we will accelerate our proximal distance algorithms by applying the algorithm map $M(\boldsymbol{x})$ given by equation (2) to the shifted point $\boldsymbol{z}_k$ of equation (10), yielding the accelerated update $\boldsymbol{x}_{k+1} = M(\boldsymbol{z}_k)$. Algorithm 1 provides a schematic of a proximal distance algorithm with Nesterov acceleration. The recent paper of Ghadimi and Lan (2015) extends Nestorov acceleration to nonconvex settings.

In ideal circumstances, one can prove linear convergence of function values in the framework of Karimi et al. (2016).

Proposition 2 *Suppose C is closed and convex and $f(\boldsymbol{x})$ is L-smooth and μ-strongly convex. Then $h_\rho(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{\rho}{2} \operatorname{dist}(\boldsymbol{x}, C)^2$ possesses a unique minimum point $\boldsymbol{y}$, and the proximal distance iterates $\boldsymbol{x}_k$ satisfy*

$$h_\rho(\boldsymbol{x}_k) - h_\rho(\boldsymbol{y}) \leq \left(1 - \frac{\mu^2}{2(L+\rho)^2}\right)^k [h_\rho(\boldsymbol{x}_0) - h_\rho(\boldsymbol{y})].$$

We now turn to convergence of the penalty function iterates as the penalty constants ρ_k tends to ∞. To simplify notation, we restrict attention to a single closed constraint set S. Let us start with a proposition requiring no convexity assumptions.

Proposition 3 *If $f(\boldsymbol{x})$ is continuous and coercive and S is compact, then the proximal distance iterates $\boldsymbol{x}_k$ are bounded and the distance to the constraint set satisfies*

$$\operatorname{dist}(\boldsymbol{x}_k, S)^2 \leq \frac{c}{\rho_k}$$

for some constant c. If in addition $f(\boldsymbol{x})$ is continuously differentiable, then

$$\operatorname{dist}(\boldsymbol{x}_k, S)^2 \leq \frac{d}{\rho_k^2}$$

for some further constant d. Similar claims hold for the solutions $\boldsymbol{y}_k$ of the penalty problem $\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \frac{\rho_k}{2} \operatorname{dist}(\boldsymbol{x}, S)^2$ except that the assumption that S is compact can be dropped.

As a corollary, if the penalty sequence ρ_k tends to ∞, then all limit points of $\boldsymbol{x}_k$ must obey the constraint. Proposition 3 puts us into position to prove the next important result.

Proposition 4 *If $f(\boldsymbol{x})$ is continuously differentiable and coercive and S is convex, then the penalty function iterates defined by $\boldsymbol{y}_k \in \operatorname{argmin}_{\boldsymbol{x}}[f(\boldsymbol{x}) + \frac{\rho_k}{2} \operatorname{dist}(\boldsymbol{x}, S)^2]$ satisfy*

$$0 \leq f(\boldsymbol{y}) - f(\boldsymbol{y}_k) \leq \frac{d + 2\sqrt{d}\|\nabla f(\boldsymbol{y})\|}{2\rho_k},$$

where $\boldsymbol{y}$ attains the constrained minimum and d is the constant identified in Proposition 3.

4. Convergence: General Case

Our strategy for addressing convergence in nonconvex problems fixes ρ and relies on Zang-will's global convergence theorem (Luenberger and Ye, 1984). This result depends in turn on the notion of a closed multi-valued map $\boldsymbol{N}(\boldsymbol{x})$. If $\boldsymbol{x}_k$ converges to $\boldsymbol{x}_\infty$ and $\boldsymbol{y}_k \in N(\boldsymbol{x}_k)$ converges to $\boldsymbol{y}_\infty$, then for $\boldsymbol{N}(\boldsymbol{x})$ to be closed, we must have $\boldsymbol{y}_\infty \in N(\boldsymbol{x}_\infty)$. The next proposition furnishes a prominent example.

Proposition 5 *If S is a closed nonempty set in $\mathbb{R}^p$, then the projection operator $P_S(\boldsymbol{x})$ is closed. Furthermore, if the sequence $\boldsymbol{x}_k$ is bounded, then the set $\cup_k P_S(\boldsymbol{x}_k)$ is bounded as well.*

ALGORITHM 1: A typical proximal distance algorithm

---

input :

$\rho_{\text{initial}} > 0$ `# an initial penalty value`

$\rho_{\text{inc}} > 1$ `# the penalty increment`

ρ_{max} `# a maximum penalty value`

K_{max} `# the maximum number of iterations`

k_ρ `# the increment frequency`

f `# the function to optimize`

P_C `# the projection onto the constraint set`

$\epsilon_{\text{loss}} > 0$ `# convergence tolerance for the loss function`

$\epsilon_{\text{dist}} > 0$ `# convergence tolerance for constraint feasibility`

output: A vector $\boldsymbol{x}^+ \approx \text{argmin}_{\boldsymbol{x}} f(\boldsymbol{x})$ `# subject to the constraint` $\boldsymbol{x} \in C$

$\rho \leftarrow \rho_{\text{initial}}$ `# Set initial penalty value`

$q_0 = q_1 = \infty$ `# Track convergence of` f

$d_0 = d_1 = \infty$ `# Track distance to constraint`

$\boldsymbol{x}_0 = \boldsymbol{x}_1 = \boldsymbol{0}$ `# Set initial iterates to origin`

`# Main algorithm loop`

for $k = 2, \ldots, K_{\text{max}}$ **do**

    $\boldsymbol{z}_k \leftarrow \boldsymbol{x}_{k-1} + \frac{k-1}{k+2}\left(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2}\right)$ `# Apply Nesterov acceleration`

    $\boldsymbol{x}_{k-2} \leftarrow \boldsymbol{x}_{k-1}$ `# Save penultimate iterate`

    $\boldsymbol{z}_{\text{proj},k} \leftarrow P_C\left(\boldsymbol{z}_k\right)$ `# Project onto constraints`

    $\boldsymbol{x}_k \leftarrow \text{prox}_{\rho f}\left(\boldsymbol{z}_{\text{proj},k}\right)$ `# Apply proximal distance update`

    $q_k \leftarrow f\left(\boldsymbol{x}_k\right)$ `# Compute new loss`

    $d_k \leftarrow \left\|\boldsymbol{x}_k - \boldsymbol{z}_{\text{proj},k}\right\|_2$ `# Compute new distance to` C

    `# Exit if converged`

    if $|q_k - q_{k-1}| < \epsilon_{\text{loss}}$ and $|d_k - d_{k-1}| < \epsilon_{\text{dist}}$ **then**

        return $\boldsymbol{x}^+ \leftarrow \boldsymbol{x}_{k+1}$

    else

        $q_{k-1} \leftarrow q_k$ `# Save current loss`

        $d_{k-1} \leftarrow d_k$ `# Save current distance to` C

        `# Update penalty` ρ `every` k_ρ `iterations`

        if $k = k_\rho$ **then**

            $\rho \leftarrow \min\left(\rho_{\text{max}}, \rho \times \rho_{\text{inc}}\right)$

            $\boldsymbol{x}_{k-1} \leftarrow \boldsymbol{x}_k$ `# Save previous iterate`

Zangwill's global convergence theorem is phrased in terms of an algorithm map $M(\boldsymbol{x})$ and a real-valued objective $h(\boldsymbol{x})$. The theorem requires a critical set Γ outside which $M(\boldsymbol{x})$ is closed. Furthermore, all iterates $\boldsymbol{x}_{k+1} \in M(\boldsymbol{x}_k)$ must fall within a compact set. Finally, the descent condition $h(\boldsymbol{y}) \leq h(\boldsymbol{x})$ should hold for all $\boldsymbol{y} \in M(\boldsymbol{x})$, with strict inequality when $\boldsymbol{x} \notin \Gamma$. If these conditions are valid, then every convergent subsequence of $\boldsymbol{x}_k$ tends to a point in Γ. In the proximal distance context, we define the complement of Γ to consist of the points $\boldsymbol{x}$ with

$$f(\boldsymbol{y}) + \frac{\rho}{2}\operatorname{dist}(\boldsymbol{y}, S)^2 \quad < \quad f(\boldsymbol{x}) + \frac{\rho}{2}\operatorname{dist}(\boldsymbol{x}, S)^2$$

for all $\boldsymbol{y} \in M(\boldsymbol{x})$. This definition plus the monotonic nature of the proximal distance algorithm

$$\boldsymbol{x}_{k+1} \quad \in \quad M(\boldsymbol{x}_k) \quad = \quad \bigcup_{\boldsymbol{z}_k \in P_S(\boldsymbol{x}_k)} \operatorname{argmin}_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{z}_k\|^2 \right]$$

force the satisfaction of Zangwill's final requirement. Note that if $f(\boldsymbol{x})$ is differentiable, then a point $\boldsymbol{x}$ belongs to Γ whenever $\boldsymbol{0} \in \nabla f(\boldsymbol{x}) + \rho\boldsymbol{x} - \rho P_S(\boldsymbol{x})$.

In general, the algorithm map $M(\boldsymbol{x})$ is multi-valued in two senses. First, for a given $\boldsymbol{z}_k \in P_S(\boldsymbol{x}_k)$, the minimum may be achieved at multiple points. This contingency is ruled out if the proximal map of $f(\boldsymbol{x})$ is unique. Second, because S may be nonconvex, the projection may be multi-valued. This sounds distressing, but the points $\boldsymbol{x}_k$ where this occurs are exceptionally rare. Accordingly, it makes no practical difference that we restrict the anchor points $\boldsymbol{z}_k$ to lie in $P_S(\boldsymbol{x}_k)$ rather than in $\operatorname{conv} P_S(\boldsymbol{x}_k)$.

Proposition 6 *If S is a closed nonempty set in $\mathbb{R}^p$, then the projection operator $P_S(\boldsymbol{x})$ is single valued except on a set of Lebesgue measure 0.*

In view of the preceding results, one can easily verify the next proposition.

Proposition 7 *The algorithm map $M(\boldsymbol{x})$ is everywhere closed.*

To apply Zangwill's global convergence theory, we must in addition prove that the iterates $\boldsymbol{x}_{k+1} = M(\boldsymbol{x}_k)$ remain within a compact set. This is true whenever the objective is coercive since the algorithm is a descent algorithm. As noted earlier, the coercivity of $f(\boldsymbol{x})$ is a sufficient condition. One can readily concoct other sufficient conditions. For example, if $f(\boldsymbol{x})$ is bounded below, say nonnegative, and S is compact, then the objective is also coercive. Indeed, if S is contained in the ball of radius r about the origin, then

$$\|\boldsymbol{x}\| \quad \leq \quad \|\boldsymbol{x} - P_S(\boldsymbol{x})\| + \|P_S(\boldsymbol{x})\| \quad \leq \quad \operatorname{dist}(\boldsymbol{x}, S) + r,$$

which proves that $\operatorname{dist}(\boldsymbol{x}, S)$ is coercive. The next proposition summarizes these findings.

Proposition 8 *If S is closed and nonempty, the objective $f(\boldsymbol{x}) + \frac{1}{2}\operatorname{dist}(\boldsymbol{x}, S)^2$ is coercive, and the proximal operator $\operatorname{prox}_{\rho^{-1}f}(\boldsymbol{x})$ is everywhere nonempty, then all limit points of the iterates $\boldsymbol{x}_{k+1} \in M(\boldsymbol{x}_k)$ of the proximal distance algorithm occur in the critical set Γ.*

This result is slightly disappointing. A limit point $\boldsymbol{x}$ could potentially exist with improvement in the objective for some but not all $\boldsymbol{y} \in \operatorname{conv} P_S(\boldsymbol{x})$. This fault is mitigated by the fact that $P_S(\boldsymbol{x})$ is almost always single valued. In common with other algorithms in nonconvex optimization, we also cannot rule out convergence to a local minimum or a saddlepoint. One can improve on Proposition 8 by assuming that the surrogates $g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k)$ are all μ-strongly convex. This is a small concession to make because ρ is typically large. If $f(\boldsymbol{x})$ is convex, then $g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k)$ is ρ-strongly convex by definition. It is also worth noting that any convex MM surrogate $g(\boldsymbol{x} \mid \boldsymbol{x}_k)$ can be made μ-strongly convex by adding the viscosity penalty $\frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{x}_k\|^2$ majorizing 0. The addition of a viscosity penalty seldom complicates finding the next iterate $\boldsymbol{x}_{n+1}$ and has little impact on the rate of convergence when $\mu > 0$ is small.

Proposition 9 *Under the μ-strongly convexity assumption on the surrogates $g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k)$, the proximal distance iterates satisfy $\lim_{k\to\infty} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| = 0$. As a consequence, the set of limit points is connected as well as closed. Furthermore, if each limit point is isolated, then the iterates converge to a critical point.*

Further progress requires even more structure. Fortunately, what we now pursue applies to generic MM algorithms. We start with the concept of a Fréchet subdifferential (Kruger, 2003). If $h(\boldsymbol{x})$ is a function mapping $\mathbb{R}^p$ into $\mathbb{R} \cup \{+\infty\}$, then its Fréchet subdifferential at $\boldsymbol{x} \in \operatorname{dom} f$ is the set

$$\partial^F h(\boldsymbol{x}) \;\; = \;\; \left\{ \boldsymbol{v} : \liminf_{\boldsymbol{y}\to\boldsymbol{x}} \frac{h(\boldsymbol{y}) - h(\boldsymbol{x}) - \boldsymbol{v}^t(\boldsymbol{y} - \boldsymbol{x})}{\|\boldsymbol{y} - \boldsymbol{x}\|} \geq 0 \right\}.$$

The set $\partial^F h(\boldsymbol{x})$ is closed, convex, and possibly empty. If $h(\boldsymbol{x})$ is convex, then $\partial^F h(\boldsymbol{x})$ reduces to its convex subdifferential. If $h(\boldsymbol{x})$ is differentiable, then $\partial^F h(\boldsymbol{x})$ reduces to its ordinary differential. At a local minimum $\boldsymbol{x}$, Fermat's rule $\boldsymbol{0} \in \partial^F h(\boldsymbol{x})$ holds.

Proposition 10 *In an MM algorithm, suppose that $h(\boldsymbol{x})$ is coercive, that the surrogates $g(\boldsymbol{x} \mid \boldsymbol{x}_k)$ are differentiable, and that the algorithm map $M(\boldsymbol{x})$ is closed. Then every limit point $\boldsymbol{z}$ of the MM sequence $\boldsymbol{x}_k$ is critical in the sense that $\boldsymbol{0} \in \partial^F(-h)(\boldsymbol{z})$.*

We will also need to invoke Łojasiewicz's inequality. This deep result depends on some rather arcane algebraic geometry (Bierstone and Milman, 1988; Bochnak et al., 2013). It applies to semialgebraic functions and their more inclusive cousins semianalytic functions and subanalytic functions. For simplicity we focus on semialgebraic functions. The class of semialgebraic subsets of $\mathbb{R}^p$ is the smallest class that:

a) contains all sets of the form $\{\boldsymbol{x} : q(\boldsymbol{x}) > 0\}$ for a polynomial $q(\boldsymbol{x})$ in p variables,

b) is closed under the formation of finite unions, finite intersections, and set complementation.

A function $a : \mathbb{R}^p \mapsto \mathbb{R}^r$ is said to be semialgebraic if its graph is a semialgebraic set of $\mathbb{R}^{p+r}$. The class of real-valued semialgebraic functions contains all polynomials $p(\boldsymbol{x})$ and all 0/1 indicators of algebraic sets. It is closed under the formation of sums, products, absolute

values, reciprocals when $a(\boldsymbol{x}) \neq 0$, nth roots when $a(\boldsymbol{x}) \geq 0$, and maxima $\max\{a(\boldsymbol{x}), b(\boldsymbol{x})\}$ and minima $\min\{a(\boldsymbol{x}), b(\boldsymbol{x})\}$. For our purposes, it is important to note that $\mathrm{dist}(\boldsymbol{x}, S)$ is a semialgebraic function whenever S is a semialgebraic set.

Łojasiewicz's inequality in its modern form (Bolte et al., 2007) requires a function $h(\boldsymbol{x})$ to be closed (lower semicontinuous) and subanalytic with a closed domain. If $\boldsymbol{z}$ is a critical point of $h(\boldsymbol{x})$, then

$$|h(\boldsymbol{x}) - h(\boldsymbol{z})|^{\theta} \quad \leq \quad c\|\boldsymbol{v}\|$$

for all $\boldsymbol{x} \in B_r(\boldsymbol{z}) \cap \mathrm{dom}\, \partial^F h$ satisfying $h(\boldsymbol{x}) > h(\boldsymbol{z})$ and all $\boldsymbol{v}$ in $\partial^F h(\boldsymbol{x})$. Here the exponent $\theta \in [0, 1)$, the radius r, and the constant c depend on $\boldsymbol{z}$. This inequality is valid for semi-algebraic functions since they are automatically subanalytic. We will apply Łojasiewicz's inequality to the limit points of an MM algorithm. The next proposition is an elaboration and expansion of known results (Attouch et al., 2010; Bolte et al., 2007; Cui et al., 2018; Kang et al., 2015; Le Thi et al., 2009).

Proposition 11 *In an MM algorithm suppose the objective $h(\boldsymbol{x})$ is coercive, continuous, and subanalytic and all surrogates $g(\boldsymbol{x} \mid \boldsymbol{x}_k)$ are continuous, μ-strongly convex, and satisfy the L-smoothness condition*

$$\|\nabla g(\boldsymbol{a} \mid \boldsymbol{x}_k) - \nabla g(\boldsymbol{b} \mid \boldsymbol{x}_k)\| \quad \leq \quad L\|\boldsymbol{a} - \boldsymbol{b}\|$$

on the compact set $\{\boldsymbol{x} : h(\boldsymbol{x}) \leq h(\boldsymbol{x}_0)\}$. Then the MM iterates $\boldsymbol{x}_{k+1} = \mathrm{argmin}_{\boldsymbol{x}}\, g(\boldsymbol{x} \mid \boldsymbol{x}_k)$ converge to a critical point.

The last proposition applies to proximal distance algorithms. The loss $f(\boldsymbol{x})$ must be subanalytic and differentiable with a locally Lipschitz gradient. Furthermore, all surrogates $g(\boldsymbol{x} \mid \boldsymbol{x}_k) = f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{y}_k\|^2$ should be coercive and μ-strongly convex. Finally, the constraints sets S_i should be subanalytic. Semialgebraic sets and functions will do. Under these conditions and regardless of how the projected points $P_{S_i}(\boldsymbol{x})$ are chosen, the MM iterates are guaranteed to converge to a critical point.

5. Examples

The following examples highlight the versatility of proximal distance algorithms in a variety of convex and nonconvex settings. Programming details matter in solving these problems. Individual programs are not necessarily long, but care must be exercised in projecting onto constraints, choosing tuning schedules, folding constraints into the domain of the loss, implementing acceleration, and declaring convergence. All of our examples are coded in the Julia programming language. Whenever possible, competing software was run in the Julia environment via the Julia module `MathProgBase` (Dunning et al., 2017; Lubin and Dunning, 2015). The sparse PCA problem relies on the software of Witten et al. (Witten et al., 2009), which is coded in R. Convergence is tested at iteration k by the two criteria

$$|f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k-1})| \quad \leq \quad \epsilon_1[|f(\boldsymbol{x}_{k-1})| + 1] \quad \text{and} \quad \mathrm{dist}(\boldsymbol{x}_k, C) \quad \leq \quad \epsilon_2,$$

where $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-4}$ are typical values. The number of iterations until convergence is about 1000 in most examples. This handicap is offset by the simplicity of each stereotyped update. Our code is available as supplementary material to this paper. Readers are encouraged to try the code and adapt it to their own examples.

5.1. Linear Programming

Two different tactics suggest themselves for constructing a proximal distance algorithm. The first tactic rolls the standard affine constraints $\boldsymbol{Ax} = \boldsymbol{b}$ into the domain of the loss function $\boldsymbol{v}^t\boldsymbol{x}$. The standard nonnegativity requirement $\boldsymbol{x} \geq \boldsymbol{0}$ is achieved by penalization. Let $\boldsymbol{x}_k$ be the current iterate and $\boldsymbol{y}_k = (\boldsymbol{x}_k)_+$ be its projection onto $\mathbb{R}_+^n$. Derivation of the proximal distance algorithm relies on the Lagrangian

$$\boldsymbol{v}^t\boldsymbol{x} + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{y}_k\|^2 + \boldsymbol{\lambda}^t(\boldsymbol{Ax} - \boldsymbol{b}).$$

One can multiply the corresponding stationarity equation

$$\boldsymbol{0} \;=\; \boldsymbol{v} + \rho(\boldsymbol{x} - \boldsymbol{y}_k) + \boldsymbol{A}^t\boldsymbol{\lambda}$$

by $\boldsymbol{A}$ and solve for the Lagrange multiplier $\boldsymbol{\lambda}$ in the form

$$\boldsymbol{\lambda} \;=\; (\boldsymbol{AA}^t)^{-1}(\rho\boldsymbol{Ay}_k - \rho\boldsymbol{b} - \boldsymbol{Av}), \tag{11}$$

assuming $\boldsymbol{A}$ has full row rank. Inserting this value into the stationarity equation gives the MM update

$$\boldsymbol{x}_{k+1} \;=\; \boldsymbol{y}_k - \frac{1}{\rho}\boldsymbol{v} - \boldsymbol{A}^-\left(\boldsymbol{Ay}_k - \boldsymbol{b} - \frac{1}{\rho}\boldsymbol{Av}\right), \tag{12}$$

where $\boldsymbol{A}^- = \boldsymbol{A}^t(\boldsymbol{AA}^t)^{-1}$ is the pseudo-inverse of $\boldsymbol{A}$.

The second tactic folds the nonnegativity constraints into the domain of the loss. Let $\boldsymbol{p}_k$ denote the projection of $\boldsymbol{x}_k$ onto the affine constraint set $\boldsymbol{Ax} = \boldsymbol{b}$. Fortunately, the surrogate function $\boldsymbol{v}^t\boldsymbol{x} + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{p}_k\|^2$ splits the parameters. Minimizing one component at a time gives the update $\boldsymbol{x}_{k+1}$ with components

$$x_{k+1,j} \;=\; \max\left\{p_{kj} - \frac{v_j}{\rho}, 0\right\}. \tag{13}$$

The projection $\boldsymbol{p}_k$ can be computed via

$$\boldsymbol{p}_k \;=\; \boldsymbol{x}_k - \boldsymbol{A}^-(\boldsymbol{Ax}_k - \boldsymbol{b}), \tag{14}$$

where $\boldsymbol{A}^-$ is again the pseudo-inverse of $\boldsymbol{A}$.

Table 1 compares the accelerated versions of these two proximal distance algorithms to two efficient solvers. The first is the open-source Splitting Cone Solver (SCS) (O'Donoghue et al., 2016), which relies on a fast implementation of ADMM. The second is the commercial Gurobi solver, which ships with implementations of both the simplex method and a barrier (interior point) method; in this example, we use its barrier algorithm. The first seven rows of the table summarize linear programs with dense data $\boldsymbol{A}$, $\boldsymbol{b}$, and $\boldsymbol{v}$. The bottom six rows rely on random sparse matrices $\boldsymbol{A}$ with sparsity level 0.01. For dense problems, the proximal distance algorithms start the penalty constant ρ at 1 and double it every 100 iterations. Because we precompute and cache the pseudoinverse $\boldsymbol{A}^-$ of $\boldsymbol{A}$, the updates (12) and (13) reduce to vector additions and matrix-vector multiplications.

Table 1: CPU times and optima for linear programming. Here m is the number of constraints, n is the number of variables, PD1 is the proximal distance algorithm over an affine domain, PD2 is the proximal distance algorithm over a nonnegative domain, SCS is the Splitting Cone Solver, and Gurobi is the Gurobi solver. After $m = 512$ the constraint matrix $\boldsymbol{A}$ is initialized to be sparse with sparsity level $s = 0.01$.

Dimensions		Optima				CPU Times (secs)			
m	n	PD1	PD2	SCS	Gurobi	PD1	PD2	SCS	Gurobi
2	4	0.2629	0.2629	0.2629	0.2629	0.0142	0.0010	0.0034	0.0038
4	8	1.0455	1.0457	1.0456	1.0455	0.0212	0.0021	0.0009	0.0011
8	16	2.4513	2.4515	2.4514	2.4513	0.0361	0.0048	0.0018	0.0029
16	32	3.4226	3.4231	3.4225	3.4223	0.0847	0.0104	0.0090	0.0036
32	64	6.2398	6.2407	6.2397	6.2398	0.1428	0.0151	0.0140	0.0055
64	128	14.671	14.674	14.671	14.671	0.2117	0.0282	0.0587	0.0088
128	256	27.116	27.125	27.116	27.116	0.3993	0.0728	0.8436	0.0335
256	512	58.501	58.512	58.494	58.494	0.7426	0.1538	2.5409	0.1954
512	1024	135.35	135.37	135.34	135.34	1.6413	0.5799	5.0648	1.7179
1024	2048	254.50	254.55	254.47	254.48	2.9541	3.2127	3.9433	0.6787
2048	4096	533.29	533.35	533.23	533.23	7.3669	17.318	25.614	5.2475
4096	8192	991.78	991.88	991.67	991.67	30.799	95.974	98.347	46.957
8192	16384	2058.8	2059.1	2058.5	2058.5	316.44	623.42	454.23	400.59

For sparse problems the proximal distance algorithms update ρ by a factor of 1.5 every 50 iterations. To avoid computing large pseudoinverses, we appeal to the LSQR variant of the conjugate gradient method (Paige and Saunders, 1982b,a) to solve the linear systems (11) and (14). The optima of all four methods agree to about 4 digits of accuracy. It is hard to declare an absolute winner in these comparisons. Gurobi and SCS clearly perform better on low-dimensional problems, but the proximal distance algorithms are competitive as dimensions increase. PD1, the proximal distance algorithm over an affine domain, tends to be more accurate than PD2. If high accuracy is not a concern, then the proximal distance algorithms are easily accelerated with a more aggressive update schedule for ρ.

5.2. Constrained Least Squares

Constrained least squares programming subsumes constrained quadratic programming. A typical quadratic program involves minimizing the quadratic $\frac{1}{2}\boldsymbol{x}^t\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{p}^t\boldsymbol{x}$ subject to $\boldsymbol{x} \in C$ for a positive definite matrix $\boldsymbol{Q}$. Quadratic programming can be reformulated as least squares by taking the Cholesky decomposition $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^t$ of $\boldsymbol{Q}$ and noting that

$$\frac{1}{2}\boldsymbol{x}^t\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{p}^t\boldsymbol{x} = \frac{1}{2}\|\boldsymbol{L}^{-1}\boldsymbol{p} - \boldsymbol{L}^t\boldsymbol{x}\|^2 - \frac{1}{2}\|\boldsymbol{L}^{-1}\boldsymbol{p}\|^2.$$

The constraint $\boldsymbol{x} \in C$ applies in both settings. It is particularly advantageous to reframe a quadratic program as a least squares problem when $\boldsymbol{Q}$ is already presented in factored

Dimensions		Optima			CPU Times		
n	p	PD	IPOPT	Gurobi	PD	IPOPT	Gurobi
16	8	4.1515	4.1515	4.1515	0.0038	0.0044	0.0010
32	16	10.8225	10.8225	10.8225	0.0036	0.0039	0.0010
64	32	29.6218	29.6218	29.6218	0.0079	0.0079	0.0019
128	64	43.2626	43.2626	43.2626	0.0101	0.0078	0.0033
256	128	111.7642	111.7642	111.7642	0.0872	0.0151	0.0136
512	256	231.6455	231.6454	231.6454	0.1119	0.0710	0.0619
1024	512	502.1276	502.1276	502.1276	0.2278	0.4013	0.2415
2048	1024	994.2447	994.2447	994.2447	1.2575	2.3346	1.1682
4096	2048	2056.8381	2056.8381	2056.8381	1.3253	15.2214	7.4971
8192	4096	4103.4611	4103.4611	4103.4611	3.0289	146.1604	49.7411
16384	8192	8295.2136	8295.2136	8295.2136	6.8739	732.1039	412.3612

Table 2: CPU times and optima for simplex-constrained least squares. Here $\boldsymbol{A} \in \mathbb{R}^{n \times p}$, PD is the proximal distance algorithm, IPOPT is the Ipopt solver, and Gurobi is the Gurobi solver. After $n = 1024$, the predictor matrix $\boldsymbol{A}$ is sparse.

form or when it is nearly singular (Bemporad, 2018). To simplify subsequent notation, we replace $\boldsymbol{L}^t$ by the rectangular matrix $\boldsymbol{A}$ and $\boldsymbol{L}^{-1}\boldsymbol{p}$ by $\boldsymbol{y}$. The key to solving constrained least squares is to express the proximal distance surrogate as

$$\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 + \frac{\rho}{2}\|\boldsymbol{x} - P_C(\boldsymbol{x}_k)\|^2 \quad = \quad \frac{1}{2}\left\|\begin{pmatrix} \boldsymbol{y} \\ \sqrt{\rho}P_C(\boldsymbol{x}_k) \end{pmatrix} - \begin{pmatrix} \boldsymbol{A} \\ \sqrt{\rho}\boldsymbol{I} \end{pmatrix}\boldsymbol{x}\right\|^2$$

as in equation (6). As noted earlier, in sparse problems the update $\boldsymbol{x}_{k+1}$ can be found by a fast stable conjugate gradient solver.

Table 2 compares the performance of the proximal distance algorithm for least squares estimation with probability-simplex constraints to the open source nonlinear interior point solver Ipopt (Wächter and Biegler, 2005, 2006) and the interior point method of Gurobi. Simplex constrained problems arise in hyperspectral imaging (Heylen et al., 2011; Keshava, 2003), portfolio optimization (Markowitz, 1952), and density estimation (Bunea et al., 2010). Test problems were generated by filling an $n \times p$ matrix $\boldsymbol{A}$ and an n-vector $\boldsymbol{y}$ with standard normal deviates. For sparse problems we set the sparsity level of $\boldsymbol{A}$ to be $10/p$. Our setup ensures that $\boldsymbol{A}$ has full rank and that the quadratic program has a solution. For the proximal distance algorithm, we start ρ at 1 and multiply it by 1.5 every 200 iterations. Table 2 suggests that the proximal distance algorithm and the interior point solvers perform equally well on small dense problems. However, in high-dimensional and low-accuracy environments, the proximal distance algorithm provides much better scalability.

5.3. Closest Kinship Matrix

In genetics studies, kinship is measured by the fraction of genes two individuals share identical by descent. For a given pedigree, the kinship coefficients for all pairs of individuals

appear as entries in a symmetric kinship matrix $\boldsymbol{Y}$. This matrix possesses three crucial properties: a) it is positive semidefinite, b) its entries are nonnegative, and c) its diagonal entries are $\frac{1}{2}$ unless some pedigree members are inbred. Inbreeding is the exception rather than the rule. Kinship matrices can be estimated empirically from single nucleotide polymorphism (SNP) data, but there is no guarantee that the three highlighted properties are satisfied. Hence, it helpful to project $\boldsymbol{Y}$ to the nearest qualifying matrix.

This projection problem is best solved by folding the positive semidefinite constraint into the domain of the Frobenius loss function $\frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2$. As we shall see, the alternative of imposing two penalties rather than one is slower and less accurate. Projection onto the constraints implied by conditions b) and c) is trivial. All diagonal entries x_{ii} of $\boldsymbol{X}$ are reset to $\frac{1}{2}$, and all off-diagonal entries x_{ij} are reset to $\max\{x_{ij}, 0\}$. If $P(\boldsymbol{X}_k)$ denotes the current projection, then the proximal distance algorithm minimizes the surrogate

$$
\begin{aligned}
g(\boldsymbol{X} \mid \boldsymbol{X}_k) &= \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \frac{\rho}{2}\|\boldsymbol{X} - P(\boldsymbol{X}_k)\|_F^2 \\
&= \frac{1+\rho}{2}\left\|\boldsymbol{X} - \frac{1}{1+\rho}\boldsymbol{Y} - \frac{\rho}{1+\rho}P(\boldsymbol{X}_k)\right\|_F^2 + c_k,
\end{aligned}
$$

where c_k is an irrelevant constant. The minimum is found by extracting the spectral decomposition $\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^t$ of $\frac{1}{1+\rho}\boldsymbol{Y} + \frac{\rho}{1+\rho}P(\boldsymbol{X}_k)$ and truncating the negative eigenvalues. This gives the update $\boldsymbol{X}_{k+1} = \boldsymbol{U}\boldsymbol{D}_+\boldsymbol{U}^t$ in obvious notation. This proximal distance algorithm and its Nesterov acceleration are simple to implement in a numerically oriented language such as Julia. The most onerous part of the calculation is clearly the repeated eigen-decompositions.

Table 3 compares three versions of the proximal distance algorithm to Dykstra's algorithm (Boyle and Dykstra, 1986). Higham proposed Dykstra's algorithm for the related problem of finding the closest correlation matrix Higham (2002). In Table 3 algorithm PD1 is the unadorned proximal distance algorithm, PD2 is the accelerated proximal distance, and PD3 is the accelerated proximal distance algorithm with the positive semidefinite constraints folded into the domain of the loss. On this demanding problem, these algorithms are comparable to Dykstra's algorithm in speed but slightly less accurate. Acceleration of the proximal distance algorithm is effective in reducing both execution time and error. Folding the positive semidefinite constraint into the domain of the loss function leads to further improvements. The data matrices $\boldsymbol{M}$ in these trials were populated by standard normal deviates and then symmetrized by averaging opposing off-diagonal entries. In algorithm PD1 we set $\rho_k = \max\{1.2^k, 2^{22}\}$. In the accelerated versions PD2 and PD3 we started ρ at 1 and multiplied it by 5 every 100 iterations. At the expense of longer compute times, better accuracy can be achieved by all three proximal distance algorithms with a less aggressive update schedule.

5.4. Projection onto a Second-Order Cone Constraint

Second-order cone programming is one of the unifying themes of convex analysis (Alizadeh and Goldfarb, 2003; Lobo et al., 1998). It revolves around conic constraints of the form $\{\boldsymbol{u} : \|\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b}\| \leq \boldsymbol{c}^t\boldsymbol{u} + d\}$. Projection of a vector $\boldsymbol{x}$ onto such a constraint is facilitated by parameter splitting. In this setting parameter splitting introduces a vector $\boldsymbol{w}$, a scalar r, and the two affine constraints $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{b}$ and $r = \boldsymbol{c}^t\boldsymbol{u} + d$. The conic constraint then

Table 3: CPU times and optima for the closest kinship matrix problem. Here the kinship matrix is $n \times n$, PD1 is the proximal distance algorithm, PD2 is the accelerated proximal distance, PD3 is the accelerated proximal distance algorithm with the positive semidefinite constraints folded into the domain of the loss, and Dykstra is Dykstra's adaptation of alternating projections. All times are in seconds.

Size	PD1		PD2		PD3		Dykstra	
n	Loss	Time	Loss	Time	Loss	Time	Loss	Time
2	1.64	0.36	1.64	0.01	1.64	0.01	1.64	0.00
4	2.86	0.10	2.86	0.01	2.86	0.01	2.86	0.00
8	18.77	0.21	18.78	0.03	18.78	0.03	18.78	0.00
16	45.10	0.84	45.12	0.18	45.12	0.12	45.12	0.02
32	169.58	4.36	169.70	0.61	169.70	0.52	169.70	0.37
64	837.85	16.77	838.44	2.90	838.43	2.63	838.42	4.32
128	3276.41	91.94	3279.44	18.00	3279.25	14.83	3279.23	19.73
256	14029.07	403.59	14045.30	89.58	14043.59	64.89	14043.46	72.79

reduces to the Lorentz cone constraint $\|\boldsymbol{w}\| \leq r$, for which projection is straightforward (Boyd and Vandenberghe, 2009). If we concatenate the parameters into the single vector

$$
\boldsymbol{y} \;=\; \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{w} \\ r \end{pmatrix}
$$

and define $L = \{\boldsymbol{y} : \|\boldsymbol{w}\| \leq r\}$ and $M = \{\boldsymbol{y} : \boldsymbol{w} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{b} \text{ and } r = \boldsymbol{c}^t\boldsymbol{u} + d\}$, then we can rephrase the problem as minimizing $\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{u}\|^2$ subject to $\boldsymbol{y} \in L \cap M$. This is a fairly typical set projection problem except that the $\boldsymbol{w}$ and r components of $\boldsymbol{y}$ are missing in the loss function.

Taking a cue from Example 5.1, we incorporate the affine constraints in the domain of the objective function. If we represent projection onto L by

$$
P\begin{pmatrix} \boldsymbol{w}_k \\ r_k \end{pmatrix} \;=\; \begin{pmatrix} \tilde{\boldsymbol{w}}_k \\ \tilde{r}_k \end{pmatrix},
$$

then the Lagrangian generated by the proximal distance algorithm amounts to

$$
\mathcal{L} \;=\; \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{u}\|^2 + \frac{\rho}{2}\left\| \begin{pmatrix} \boldsymbol{w} - \tilde{\boldsymbol{w}}_k \\ r - \tilde{r}_k \end{pmatrix} \right\|^2 + \boldsymbol{\lambda}^t(\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b} - \boldsymbol{w}) + \theta(\boldsymbol{c}^t\boldsymbol{u} + d - r).
$$

This gives rise to a system of three stationarity equations

$$
\begin{align}
\boldsymbol{0} &= \boldsymbol{u} - \boldsymbol{x} + \boldsymbol{A}^t\boldsymbol{\lambda} + \theta\boldsymbol{c} \tag{15} \\
\boldsymbol{0} &= \rho(\boldsymbol{w} - \tilde{\boldsymbol{w}}_k) - \boldsymbol{\lambda} \tag{16} \\
0 &= \rho(r - \tilde{r}_k) - \theta. \tag{17}
\end{align}
$$

Solving for the multipliers $\boldsymbol{\lambda}$ and θ in equations (16) and (17) and substituting their values in equation (15) yield

$$
\begin{aligned}
\mathbf{0} & = \boldsymbol{u} - \boldsymbol{x} + \rho \boldsymbol{A}^t (\boldsymbol{w} - \tilde{\boldsymbol{w}}_k) + \rho (r - \tilde{r}_k) \boldsymbol{c} \\
& = \boldsymbol{u} - \boldsymbol{x} + \rho \boldsymbol{A}^t (\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b} - \tilde{\boldsymbol{w}}_k) + \rho (\boldsymbol{c}^t \boldsymbol{u} + d - \tilde{r}_k) \boldsymbol{c}.
\end{aligned}
$$

This leads to the MM update

$$
\boldsymbol{u}_{k+1} = (\rho^{-1} \boldsymbol{I} + \boldsymbol{A}^t \boldsymbol{A} + \boldsymbol{c}\boldsymbol{c}^t)^{-1} [\rho^{-1} \boldsymbol{x} + \boldsymbol{A}^t (\tilde{\boldsymbol{w}}_k - \boldsymbol{b}) + (\tilde{r}_k - d)\boldsymbol{c}]. \tag{18}
$$

The updates $\boldsymbol{w}_{k+1} = \boldsymbol{A}\boldsymbol{u}_{k+1} + \boldsymbol{b}$ and $r_{k+1} = \boldsymbol{c}^t \boldsymbol{u}_{k+1} + d$ follow from the constraints.

Table 4 compares the proximal distance algorithm to SCS and Gurobi. Echoing previous examples, we tailor the update schedule for ρ differently for dense and sparse problems. Dense problems converge quickly and accurately when we set $\rho_0 = 1$ and double ρ every 100 iterations. Sparse problems require a greater range and faster updates of ρ, so we set $\rho_0 = 0.01$ and then multiply ρ by 2.5 every 10 iterations. For dense problems, it is clearly advantageous to cache the spectral decomposition of $\boldsymbol{A}^t \boldsymbol{A} + \boldsymbol{c}\boldsymbol{c}^t$ as suggested in Example 5.2. In this regime, the proximal distance algorithm is as accurate as Gurobi and nearly as fast. SCS is comparable to Gurobi in speed but notably less accurate.

With a large sparse constraint matrix $\boldsymbol{A}$, extraction of its spectral decomposition becomes prohibitive. If we let $\boldsymbol{E} = (\rho^{-1/2}\boldsymbol{I} \ \boldsymbol{A}^t \ \boldsymbol{c})$, then we must solve a linear system of equations defined by the Gramian matrix $\boldsymbol{G} = \boldsymbol{E}\boldsymbol{E}^t$. There are three reasonable options for solving this system. The first relies on computing and caching a sparse Cholesky decomposition of $\boldsymbol{G}$. The second computes the QR decomposition of the sparse matrix $\boldsymbol{E}$. The R part of the QR decomposition coincides with the Cholesky factor. Unfortunately, every time ρ changes, the Cholesky or QR decomposition must be redone. The third option is the conjugate gradient algorithm. In our experience the QR decomposition offers superior stability and accuracy. When $\boldsymbol{E}$ is very sparse, the QR decomposition is often much faster than the Cholesky decomposition because it avoids forming the dense matrix $\boldsymbol{A}^t \boldsymbol{A}$. Even when only 5% of the entries of $\boldsymbol{A}$ are nonzero, 90% of the entries of $\boldsymbol{A}^t \boldsymbol{A}$ can be nonzero. If exquisite accuracy is not a concern, then the conjugate gradient method provides the fastest update. Table 4 reflects this choice.

5.5. Copositive Matrices

A symmetric matrix $\boldsymbol{M}$ is copositive if its associated quadratic form $\boldsymbol{x}^t \boldsymbol{M} \boldsymbol{x}$ is nonnegative for all $\boldsymbol{x} \geq \mathbf{0}$. Copositive matrices find applications in numerous branches of the mathematical sciences (Berman and Plemmons, 1994). All positive semidefinite matrices and all matrices with nonnegative entries are copositive. The variational index

$$
\mu(\boldsymbol{M}) = \min_{\|\boldsymbol{x}\|=1, \, \boldsymbol{x} \geq \mathbf{0}} \boldsymbol{x}^t \boldsymbol{M} \boldsymbol{x}
$$

is one key to understanding copositive matrices (Hiriart-Urruty and Seeger, 2010). The constraint set S is the intersection of the unit sphere and the nonnegative cone $\mathbb{R}_+^n$. Projection of an external point $\boldsymbol{y}$ onto S splits into three cases. When all components of $\boldsymbol{y}$ are negative, then $P_S(\boldsymbol{y}) = \boldsymbol{e}_i$, where y_i is the least negative component of $\boldsymbol{y}$, and $\boldsymbol{e}_i$ is the standard unit

Table 4: CPU times and optima for the second-order cone projection. Here m is the number of constraints, n is the number of variables, PD is the accelerated proximal distance algorithm, SCS is the Splitting Cone Solver, and Gurobi is the Gurobi solver. After $m = 512$ the constraint matrix $\boldsymbol{A}$ is initialized with sparsity level 0.01.

Dimensions		Optima			CPU Seconds		
m	n	PD	SCS	Gurobi	PD	SCS	Gurobi
2	4	0.10598	0.10607	0.10598	0.0043	0.0103	0.0026
4	8	0.00000	0.00000	0.00000	0.0003	0.0009	0.0022
8	16	0.88988	0.88991	0.88988	0.0557	0.0011	0.0027
16	32	2.16514	2.16520	2.16514	0.0725	0.0012	0.0040
32	64	3.03855	3.03864	3.03853	0.0952	0.0019	0.0094
64	128	4.86894	4.86962	4.86895	0.1225	0.0065	0.0403
128	256	10.5863	10.5843	10.5863	0.1975	0.0810	0.0868
256	512	31.1039	31.0965	31.1039	0.5463	0.3995	0.3405
512	1024	27.0483	27.0475	27.0483	3.7667	1.6692	2.0189
1024	2048	1.45578	1.45569	1.45569	0.5352	0.3691	1.5489
2048	4096	2.22936	2.22930	2.22921	1.0845	2.4531	5.5521
4096	8192	1.72306	1.72202	1.72209	3.1404	17.272	15.204
8192	16384	5.36191	5.36116	5.36144	13.979	133.25	88.024

vector along coordinate direction i. The origin $\boldsymbol{0}$ is equidistant from all points of S. If any component of $\boldsymbol{y}$ is positive, then the projection is constructed by setting the negative components of $\boldsymbol{y}$ equal to 0, and standardizing the truncated version of $\boldsymbol{y}$ to have Euclidean norm 1.

As a test case for the proximal distance algorithm, consider the Horn matrix (Hall and Newman, 1963)

$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 \end{bmatrix}.$$

The value $\mu(\boldsymbol{M}) = 0$ is attained for the vectors $\frac{1}{\sqrt{2}}(1,1,0,0,0)^t$, $\frac{1}{\sqrt{6}}(1,2,1,0,0)^t$, and equivalent vectors with their entries permuted. Matrices in higher dimensions with the same Horn pattern of 1's and -1's are copositive as well (Johnson and Reams, 2008). A Horn matrix of odd dimension cannot be written as a positive semidefinite matrix, a nonnegative matrix, or a sum of two such matrices.

The proximal distance algorithm minimizes the criterion

$$g(\boldsymbol{x} \mid \boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}^t\boldsymbol{M}\boldsymbol{x} + \frac{\rho}{2}\|\boldsymbol{x} - P_S(\boldsymbol{x}_k)\|^2$$

21

Table 5: CPU times (seconds) and optima for approximating the Horn variational index of a Horn matrix. Here n is the size of Horn matrix, PD is the proximal distance algorithm, aPD is the accelerated proximal distance algorithm, and Mosek is the Mosek solver.

Dimension	Optima			CPU Seconds		
n	PD	aPD	Mosek	PD	aPD	Mosek
4	0.000000	0.000000	feasible	0.5555	0.0124	2.7744
5	0.000000	0.000000	infeasible	0.0039	0.0086	0.0276
8	0.000021	0.000000	feasible	0.0059	0.0083	0.0050
9	0.000045	0.000000	infeasible	0.0055	0.0072	0.0082
16	0.000377	0.000001	feasible	0.0204	0.0237	0.0185
17	0.000441	0.000001	infeasible	0.0204	0.0378	0.0175
32	0.001610	0.000007	feasible	0.0288	0.0288	0.1211
33	0.002357	0.000009	infeasible	0.0242	0.0346	0.1294
64	0.054195	0.000026	feasible	0.0415	0.0494	3.6284
65	0.006985	0.000026	infeasible	0.0431	0.0551	2.7862

and generates the updates

$$\boldsymbol{x}_{k+1} \;=\; (\boldsymbol{M} + \rho\boldsymbol{I})^{-1}\rho P_S(\boldsymbol{x}_k).$$

It takes a gentle tuning schedule to get decent results. The choice $\rho_k = 1.2^k$ converges in 600 to 700 iterations from random starting points and reliably yields objective values below 10^{-5} for Horn matrices. The computational burden per iteration is significantly eased by exploiting the cached spectral decomposition of $\boldsymbol{M}$. Table 5 compares the performance of the proximal distance algorithm to the Mosek solver on a range of Horn matrices. Mosek uses semidefinite programming to decide whether $\boldsymbol{M}$ can be decomposed into a sum of a positive semidefinite matrix and a nonnegative matrix. If not, Mosek declares the problem infeasible. Nesterov acceleration improves the final loss for the proximal distance algorithm, but it does not decrease overall computing time.

Testing for copositivity is challenging because neither the loss function nor the constraint set is convex. The proximal distance algorithm offers a fast screening device for checking whether a matrix is copositive. On random 1000×1000 symmetric matrices $\boldsymbol{M}$, the method invariably returns a negative index in less than two seconds of computing time. Because the vast majority of symmetric matrices are not copositive, accurate estimation of the minimum is not required. Table 6 summarizes a few random trials with lower-dimensional symmetric matrices. In higher dimensions, Mosek becomes non-competitive, and Nesterov acceleration is of dubious value.

5.6. Linear Complementarity Problem

The linear complementarity problem (Murty and Yu, 1988) consists of finding vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ with nonnegative components such that $\boldsymbol{x}^t\boldsymbol{y} = 0$ and $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ for a given square

Table 6: CPU times and optima for testing the copositivity of random symmetric matrices. Here n is the size of matrix, PD is the proximal distance algorithm, aPD is the accelerated proximal distance algorithm, and Mosek is the Mosek solver.

Dimension	Optima			CPU Seconds		
n	PD	aPD	Mosek	PD	aPD	Mosek
4	-0.391552	-0.391561	infeasible	0.0029	0.0031	0.0024
8	-0.911140	-2.050316	infeasible	0.0037	0.0044	0.0045
16	-1.680697	-1.680930	infeasible	0.0199	0.0272	0.0062
32	-2.334520	-2.510781	infeasible	0.0261	0.0242	0.0441
64	-3.821927	-3.628060	infeasible	0.0393	0.0437	0.6559
128	-5.473609	-5.475879	infeasible	0.0792	0.0798	38.3919
256	-7.956365	-7.551814	infeasible	0.1632	0.1797	456.1500

matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$. The natural loss function is $\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{Ax} - \boldsymbol{b}\|^2$. To project a vector pair $(\boldsymbol{u}, \boldsymbol{v})$ onto the nonconvex constraint set, one considers each component pair (u_i, v_i) in turn. If $u_i \geq \max\{v_i, 0\}$, then the nearest pair $(\boldsymbol{x}, \boldsymbol{y})$ has components $(x_i, y_i) = (u_i, 0)$. If $v_i \geq \max\{u_i, 0\}$, then the nearest pair has components $(x_i, y_i) = (0, v_i)$. Otherwise, $(x_i, y_i) = (0, 0)$. At each iteration the proximal distance algorithm minimizes the criterion

$$\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{Ax} - \boldsymbol{b}\|^2 + \frac{\rho}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}_k\|^2 + \frac{\rho}{2}\|\boldsymbol{y} - \tilde{\boldsymbol{y}}_k\|^2,$$

where $(\tilde{\boldsymbol{x}}_k, \tilde{\boldsymbol{y}}_k)$ is the projection of $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ onto the constraint set. The stationarity equations become

$$\begin{aligned} \boldsymbol{0} &= -\boldsymbol{A}^t(\boldsymbol{y} - \boldsymbol{Ax} - \boldsymbol{b}) + \rho(\boldsymbol{x} - \tilde{\boldsymbol{x}}_k) \\ \boldsymbol{0} &= \boldsymbol{y} - \boldsymbol{Ax} - \boldsymbol{b} + \rho(\boldsymbol{y} - \tilde{\boldsymbol{y}}_k). \end{aligned}$$

Substituting the value of $\boldsymbol{y}$ from the second equation into the first equation leads to the updates

$$\begin{aligned} \boldsymbol{x}_{k+1} &= [(1+\rho)\boldsymbol{I} + \boldsymbol{A}^t\boldsymbol{A}]^{-1}[\boldsymbol{A}^t(\tilde{\boldsymbol{y}}_k - \boldsymbol{b}) + (1+\rho)\tilde{\boldsymbol{x}}_k] \qquad (19) \\ \boldsymbol{y}_{k+1} &= \frac{1}{1+\rho}(\boldsymbol{Ax}_{k+1} + \boldsymbol{b}) + \frac{\rho}{1+\rho}\tilde{\boldsymbol{y}}_k. \end{aligned}$$

The linear system (19) can be solved in low to moderate dimensions by computing and caching the spectral decomposition of $\boldsymbol{A}^t\boldsymbol{A}$ and in high dimensions by the conjugate gradient method. Table 7 compares the performance of the proximal gradient algorithm to the Gurobi solver on some randomly generated problems.

5.7. Sparse Principal Components Analysis

Let $\boldsymbol{X}$ be an $n \times p$ data matrix gathered on n cases and p predictors. Assume the columns of $\boldsymbol{X}$ are centered to have mean 0. Principal component analysis (PCA) (Hotelling, 1933;

Table 7: CPU times (seconds) and optima for the linear complementarity problem with randomly generated data. Here n is the size of matrix, PD is the accelerated proximal distance algorithm, and Gurobi is the Gurobi solver.

Dimension	Optima		CPU Seconds	
n	PD	Mosek	PD	Mosek
4	0.000000	0.000000	0.0230	0.0266
8	0.000000	0.000000	0.0062	0.0079
16	0.000000	0.000000	0.0269	0.0052
32	0.000000	0.000000	0.0996	0.4303
64	0.000074	0.000000	2.6846	360.5183

Pearson, 1901) operates on the sample covariance matrix $\boldsymbol{S} = \frac{1}{n}\boldsymbol{X}^t\boldsymbol{X}$. Here we formulate a proximal distance algorithm for sparse PCA (SPCA), which has attracted substantial interest in the machine learning community (Berthet and Rigollet, 2013b,a; D'Aspremont et al., 2007; Johnstone and Lu, 2009; Journée et al., 2010; Witten et al., 2009; Zou et al., 2006). According to a result of Ky Fan (Fan, 1949), the first q principal components (PCs) $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_q$ can be extracted by maximizing the function $\mathrm{tr}(\boldsymbol{U}^t\boldsymbol{S}\boldsymbol{U})$ subject to the matrix constraint $\boldsymbol{U}^t\boldsymbol{U} = \boldsymbol{I}_q$, where $\boldsymbol{u}_i$ is the ith column of the $p \times q$ matrix $\boldsymbol{U}$. This constraint set is called a Stiefel manifold. One can impose sparsity by insisting that any given column $\boldsymbol{u}_i$ have at most r nonzero entries. Alternatively, one can require the entire matrix $\boldsymbol{U}$ to have at most r nonzero entries. The latter choice permits sparsity to be distributed non-uniformly across columns.

Extraction of sparse PCs is difficult for three reasons. First, the Stiefel manifold M_q and both sparsity sets are nonconvex. Second, the objective function is concave rather than convex. Third, there is no simple formula or algorithm for projecting onto the intersection of the two constraint sets. Fortunately, it is straightforward to project onto each separately. Let $P_{M_q}(\boldsymbol{U})$ denote the projection of $\boldsymbol{U}$ onto the Stiefel manifold. It is well known that $P_{M_q}(\boldsymbol{U})$ can be calculated by extracting a partial singular value decomposition $\boldsymbol{U} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^t$ of $\boldsymbol{U}$ and setting $P_{M_q}(\boldsymbol{U}) = \boldsymbol{V}\boldsymbol{W}^t$ (Golub and Van Loan, 2012). Here $\boldsymbol{V}$ and $\boldsymbol{W}$ are orthogonal matrices of dimension $p \times q$ and $q \times q$, respectively, and $\boldsymbol{\Sigma}$ is a diagonal matrix of dimension $q \times q$. Let $P_{S_r}(\boldsymbol{U})$ denote the projection of $\boldsymbol{U}$ onto the sparsity set

$$S_r = \{\boldsymbol{V} : v_{ij} \neq 0 \text{ for at most } r \text{ entries of each column } \boldsymbol{v}_i\}.$$

Because $P_{S_r}(\boldsymbol{U})$ operates column by column, it suffices to project each column vector $\boldsymbol{u}_i$ to sparsity. This entails nothing more than sorting the entries of $\boldsymbol{u}_i$ by magnitude, saving the r largest, and sending the remaining $p - r$ entries to 0. If the entire matrix $\boldsymbol{U}$ must have at most r nonzero entries, then $\boldsymbol{U}$ can be treated as a concatenated vector during projection.

The key to a good algorithm is to incorporate the Stiefel constraints into the domain of the objective function (Kiers, 1990; Kiers and ten Berge, 1992) and the sparsity constraints into the distance penalty. Thus, we propose decreasing the criterion

$$f(\boldsymbol{U}) = -\frac{1}{2}\mathrm{tr}(\boldsymbol{U}^t\boldsymbol{S}\boldsymbol{U}) + \frac{\rho}{2}\mathrm{dist}(\boldsymbol{U}, S_r)^2.$$

at each iteration subject to the Stiefel constraints. The loss can be majorized via

$$
\begin{aligned}
-\frac{1}{2}\operatorname{tr}(\boldsymbol{U}^t\boldsymbol{S}\boldsymbol{U}) &= -\frac{1}{2}\operatorname{tr}[(\boldsymbol{U}-\boldsymbol{U}_k)^t\boldsymbol{S}(\boldsymbol{U}-\boldsymbol{U}_k)] - \operatorname{tr}(\boldsymbol{U}^t\boldsymbol{S}\boldsymbol{U}_k) + \frac{1}{2}\operatorname{tr}(\boldsymbol{U}_k^t\boldsymbol{S}\boldsymbol{U}_k) \\
&\leq -\operatorname{tr}(\boldsymbol{U}^t\boldsymbol{S}\boldsymbol{U}_k) + \frac{1}{2}\operatorname{tr}(\boldsymbol{U}_k^t\boldsymbol{S}\boldsymbol{U}_k)
\end{aligned}
$$

because $\boldsymbol{S}$ is positive semidefinite. The penalty is majorized by

$$
\frac{\rho}{2}\operatorname{dist}(\boldsymbol{U}, S_r)^2 \leq -\rho\operatorname{tr}[\boldsymbol{U}^t P_{S_r}(\boldsymbol{U}_k)] + c_k
$$

up to an irrelevant constant c_k since the squared Frobenius norm satisfies the relation $\|\boldsymbol{U}^t\boldsymbol{U}\|_F^2 = q$ on the Stiefel manifold. It now follows that $f(\boldsymbol{U})$ is majorized by

$$
\frac{1}{2}\|\boldsymbol{U} - \boldsymbol{S}\boldsymbol{U}_k - \rho P_{S_r}(\boldsymbol{U}_k)\|_F^2
$$

up to an irrelevant constant. Accordingly, the Stiefel projection

$$
\boldsymbol{U}_{k+1} = P_{M_q}[\boldsymbol{S}\boldsymbol{U}_k + \rho P_{S_r}(\boldsymbol{U}_k)]
$$

provides the next MM iterate.

Figures 1 and 2 compare the proximal distance algorithm to the SPC function from the R package `PMA` (Witten et al., 2009). The breast cancer data from `PMA` provide the data matrix $\boldsymbol{X}$. The data consist of $p = 19,672$ RNA measurements on $n = 89$ patients. The two figures show computation times and the proportion of variance explained (PVE) by the $p \times q$ loading matrix $\boldsymbol{U}$. For sparse PCA, PVE is defined as $\operatorname{tr}(\boldsymbol{X}_q^t\boldsymbol{X}_q)/\operatorname{tr}(\boldsymbol{X}^t\boldsymbol{X})$, where $\boldsymbol{X}_q = \boldsymbol{X}\boldsymbol{U}(\boldsymbol{U}^t\boldsymbol{U})^{-1}\boldsymbol{U}^t$ (Shen and Huang, 2008). When the loading vectors of $\boldsymbol{U}$ are orthogonal, this criterion reduces to the familiar definition $\operatorname{tr}(\boldsymbol{U}^t\boldsymbol{X}^t\boldsymbol{X}\boldsymbol{U})/\operatorname{tr}(\boldsymbol{X}^t\boldsymbol{X})$ of PVE for ordinary PCA. The proximal distance algorithm enforces either matrix-wise or column-wise sparsity. In contrast, SPC enforces only column-wise sparsity via the constraint $\|\boldsymbol{u}_i\|_1 \leq c$ for each column $\boldsymbol{u}_i$ of $\boldsymbol{U}$. We take $c = 8$. The number of nonzeroes per loading vector output by SPC dictates the sparsity level for the column-wise version of the proximal distance algorithm. Summing these counts across all columns dictates the sparsity level for the matrix version of the proximal distance algorithm.

Figures 1 and 2 demonstrate the superior PVE and computational speed of both proximal distance algorithms versus SPC. The type of projection does not appear to affect the computational performance of the proximal distance algorithm, as both versions scale equally well with q. However, the matrix projection, which permits the algorithm to more freely assign nonzeroes to the loadings, attains better PVE than the more restrictive column-wise projection. For both variants of the proximal distance algorithm, Nesterov acceleration improves both fitting accuracy and computational speed, especially as the number of PCs q increases.

6. Discussion

The proximal distance algorithm applies to a host of problems. In addition to the linear and quadratic programming examples considered here, our previous paper (Lange and Keys,
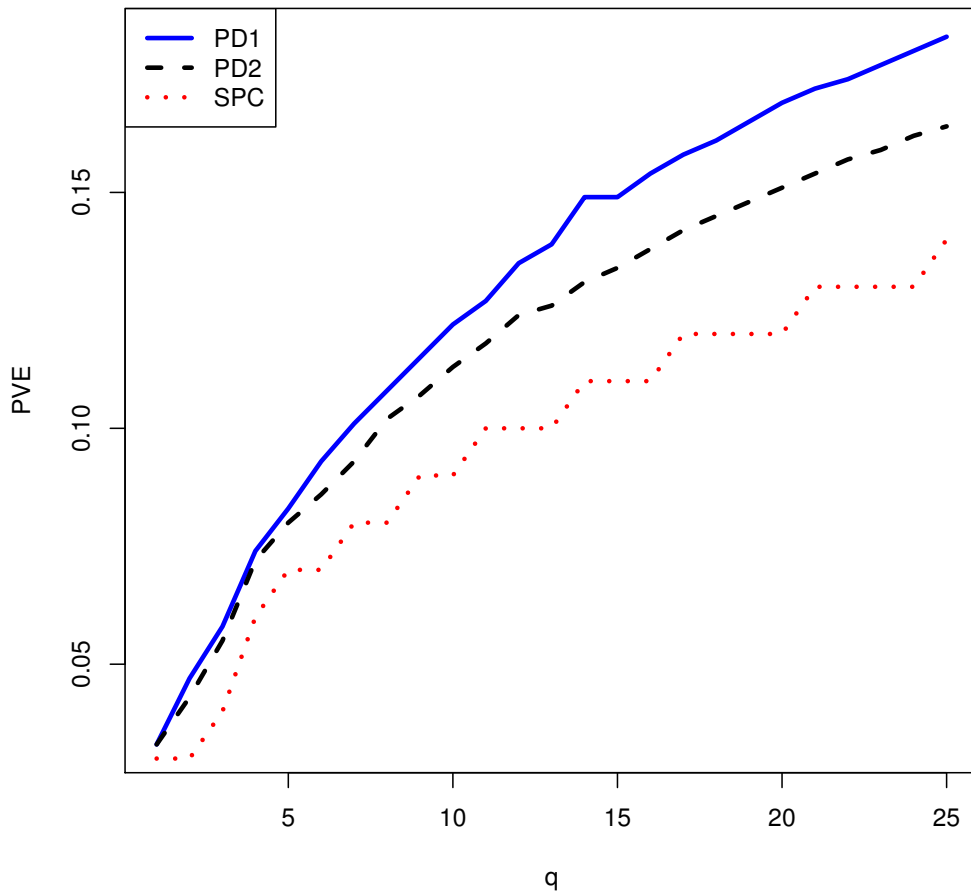
Figure 1: Proportion of variance explained by q PCs for each algorithm. Here PD1 is the accelerated proximal distance algorithm enforcing matrix sparsity, PD2 is the accelerated proximal distance algorithm enforcing column-wise sparsity, and SPC is the orthogonal sparse PCA method from `PMA`.
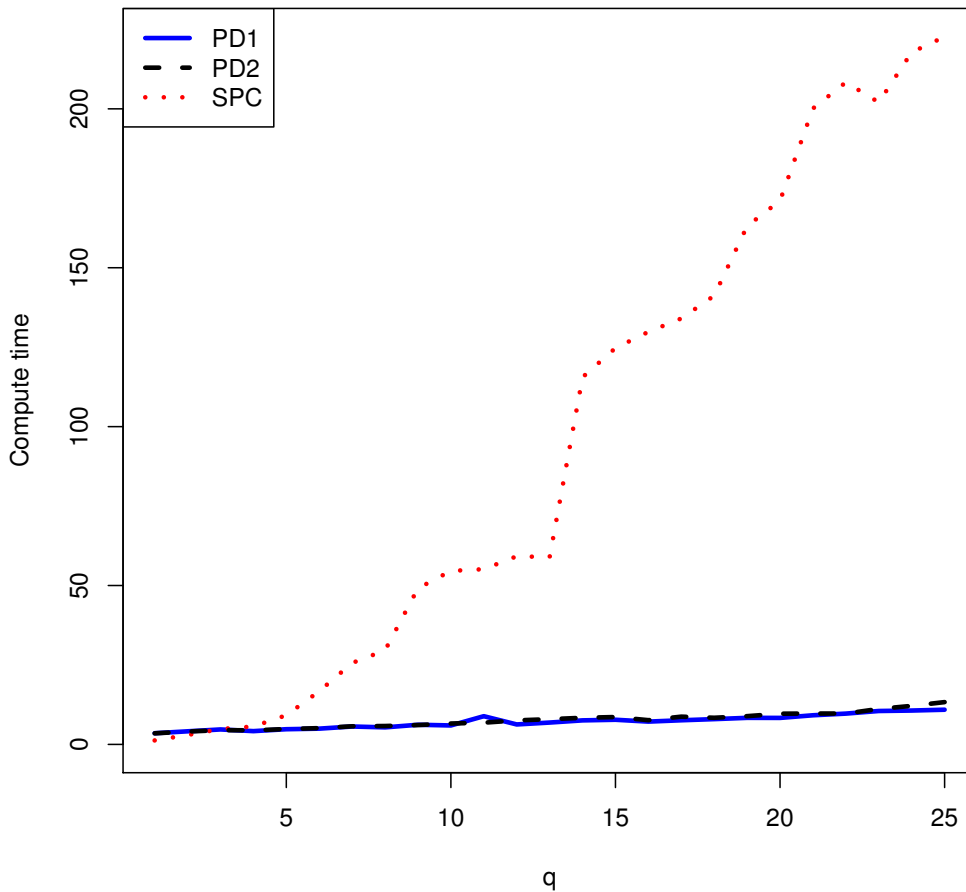
Figure 2: Computation times for q PCs for each algorithm. Here PD1 is the accelerated proximal distance algorithm enforcing matrix sparsity, PD2 is the accelerated proximal distance algorithm enforcing column-wise sparsity, and SPC is the orthogonal sparse PCA method from `PMA`.

2014) derives and tests algorithms for binary piecewise-linear programming, ℓ_0 regression, matrix completion (Cai et al., 2010; Candès and Tao, 2010; Chen et al., 2012; Mazumder et al., 2010), and sparse precision matrix estimation (Friedman et al., 2008). Other potential applications immediately come to mind. An integer linear program in standard form can be expressed as minimizing $\boldsymbol{c}^t \boldsymbol{x}$ subject to $\boldsymbol{Ax} + \boldsymbol{s} = \boldsymbol{b}$, $\boldsymbol{s} \geq \boldsymbol{0}$, and $\boldsymbol{x} \in \mathbb{Z}^p$. The latter two constraints can be combined in a single constraint for which projection is trivial. The affine constraints should be folded into the domain of the objective. Integer programming is NP hard, so that the proximal distance algorithm just sketched is merely heuristic. Integer linear programming includes traditional NP hard problems such as the traveling salesman problem, the vertex cover problem, set packing, and Boolean satisfiability. It will be interesting to see if the proximal distance principle is competitive in meeting these challenges. Our experience with the closest lattice point problem (Agrell et al., 2002) and the eight queens problem suggests that the proximal distance algorithm can be too greedy for hard combinatorial problems. The nonconvex problems solved in this paper are in some vague sense easy combinatorial problems.

The behavior of a proximal distance algorithm depends critically on a sensible tuning schedule for increasing ρ. Starting ρ too high puts too much stress on satisfying the constraints. Incrementing ρ too quickly causes the algorithm to veer off the solution path guaranteed by the penalty method. Given the chance of roundoff error even with double precision arithmetic, it is unwise to take ρ all the way to ∞. Trial and error can help in deciding whether a given class of problems will benefit from an aggressive update schedule and strict or loose convergence criteria. In problems with little curvature such as linear programming, more conservative updates are probably prudent. The linear programming, closest kinship matrix, and SPCA problems document the value of folding constraints into the domain of the loss. In the same spirit it is wise to minimize the number of constraints. A single penalty for projecting onto the intersection of two constraint sets is almost always preferable to the sum of two penalties for their separate projections. Exceptions to this rule obviously occur when projection onto the intersection is intractable. The integer linear programming problem mentioned previously illustrates these ideas.

Our earlier proximal distance algorithms ignored acceleration. In many cases the solutions produced had very low accuracy. The realization that convex proximal distance algorithms can be phrased as proximal gradient algorithms convinced us to try Nesterov acceleration. We now do this routinely on the subproblems with ρ fixed. This typically forces tighter path following and a reduction in overall computing times. Our examples generally bear out the contention that Nesterov acceleration is useful in nonconvex problems (Ghadimi and Lan, 2015). It is noteworthy that the value of acceleration often lies in improving the quality of a solution as much as it does in increasing the rate of convergence. Of course, acceleration cannot prevent convergence to an inferior local minimum.

On both convex and nonconvex problems, proximal distance algorithms enjoy global convergence guarantees. On nonconvex problems, one must confine attention to subanalytic sets and subanalytic functions. This minor restriction is not a handicap in practice. Determining local convergence rates is a more vexing issue. For convex problems, we review existing theory for a fixed penalty constant ρ. The classical results buttress an $O(\rho k^{-1})$ sublinear rate for general convex problems. Better results require restrictive smoothness assumptions on both the objective function and the constraint sets. For instance, when $f(\boldsymbol{x})$ is L-smooth

and strongly convex, linear convergence can be demonstrated. When $f(\boldsymbol{x})$ equals a difference of convex functions, proximal distance algorithms reduce to concave-convex programming. Le Thi et al. (2009) attack convergence in this setting.

We hope readers will sense the potential of the proximal distance principle. This simple idea offers insight into many existing algorithms and a straightforward path in devising new ones. Effective proximal and projection operators usually spell the difference between success and failure. The number and variety of such operators is expanding quickly as the field of optimization relinquishes it fixation on convexity. The current paper research leaves many open questions about tuning schedules, rates of convergence, and acceleration in the face of nonconvexity. We welcome the contributions of other mathematical scientists in unraveling these mysteries and in inventing new proximal distance algorithms.

Acknowledgments

Appendix A. Proofs of the Stated Propositions

A.1. Proposition 1

We first observe that the surrogate function $g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k)$ is ρ-strongly convex. Consequently, the stationarity condition $\boldsymbol{0} \in \partial g_\rho(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$ implies

$$g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) \;\geq\; g_\rho(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{x}_{k+1}\|^2 \tag{20}$$

for all $\boldsymbol{x}$. In the notation (9), the difference

$$d_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) \;=\; g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) - h_\rho(\boldsymbol{x}) \;=\; \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{y}_k\|^2 - \frac{\rho}{2}\sum_{i=1}^m \alpha_i \operatorname{dist}(\boldsymbol{x}, C_i)^2$$

is ρ-smooth because

$$\nabla d_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) \;=\; \rho(\boldsymbol{x} - \boldsymbol{y}_k) - \rho\sum_{i=1}^m \alpha_i[\boldsymbol{x} - P_{C_i}(\boldsymbol{x})] \;=\; \rho\sum_{i=1}^m \alpha_i P_{C_i}(\boldsymbol{x}) - \rho\boldsymbol{y}_k.$$

The tangency conditions $d_\rho(\boldsymbol{x}_k \mid \boldsymbol{x}_k) = 0$ and $\nabla d_\rho(\boldsymbol{x}_k \mid \boldsymbol{x}_k) = \boldsymbol{0}$ therefore yield

$$d_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) \;\leq\; d_\rho(\boldsymbol{x}_k \mid \boldsymbol{x}_k) + \nabla d_\rho(\boldsymbol{x}_k)^t(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{x}_k\|^2 \;=\; \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{x}_k\|^2 \tag{21}$$

for all $\boldsymbol{x}$. Combining inequalities (20) and (21) gives

$$
\begin{aligned}
h_\rho(\boldsymbol{x}_{k+1}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{x}_{k+1}\|^2 \;&\leq\; g_\rho(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{x}_{k+1}\|^2 \\
&\leq\; g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) \\
&=\; h_\rho(\boldsymbol{x}) + d_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) \\
&\leq\; h_\rho(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{x}_k\|^2.
\end{aligned}
$$

Adding the result

$$
h_\rho(\boldsymbol{x}_{k+1}) - h_\rho(\boldsymbol{x}) \;\leq\; \frac{\rho}{2}\Big(\|\boldsymbol{x} - \boldsymbol{x}_k\|^2 - \|\boldsymbol{x} - \boldsymbol{x}_{k+1}\|^2\Big)
$$

over k and invoking the descent property $h_\rho(\boldsymbol{x}_{k+1}) \leq h_\rho(\boldsymbol{x}_k)$ produce the error bound

$$
h_\rho(\boldsymbol{x}_{k+1}) - h_\rho(\boldsymbol{x}) \;\leq\; \frac{\rho}{2(k+1)}\Big(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 - \|\boldsymbol{x} - \boldsymbol{x}_{k+1}\|^2\Big) \;\leq\; \frac{\rho}{2(k+1)}\|\boldsymbol{x} - \boldsymbol{x}_0\|^2.
$$

Setting $\boldsymbol{x}$ equal to a minimal point $\boldsymbol{z}$ gives the stated result. ∎

A.2. Proposition 2

The existence and uniqueness of $\boldsymbol{y}$ are obvious. The remainder of the proof hinges on the assumptions that $h_\rho(\boldsymbol{x})$ is μ-strongly convex and the surrogate $g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k)$ is $L + \rho$ smooth. The latter assumption yields

$$
\begin{aligned}
h_\rho(\boldsymbol{x}) - h_\rho(\boldsymbol{y}) \;&\leq\; g_\rho(\boldsymbol{x} \mid \boldsymbol{y}) - g_\rho(\boldsymbol{y} \mid \boldsymbol{y}) \\
&\leq\; \nabla g_\rho(\boldsymbol{y} \mid \boldsymbol{y})^t(\boldsymbol{x} - \boldsymbol{y}) + \frac{L + \rho}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 \qquad (22) \\
&=\; \frac{L + \rho}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2.
\end{aligned}
$$

The strong convexity condition $h_\rho(\boldsymbol{y}) - h_\rho(\boldsymbol{x}) \geq \nabla h_\rho(\boldsymbol{x})^t(\boldsymbol{y} - \boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2$ implies

$$
\|\nabla h_\rho(\boldsymbol{x})\| \cdot \|\boldsymbol{y} - \boldsymbol{x}\| \;\geq\; -\nabla h_\rho(\boldsymbol{x})^t(\boldsymbol{y} - \boldsymbol{x}) \;\geq\; \frac{\mu}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2.
$$

It follows that $\|\nabla h_\rho(\boldsymbol{x})\| \geq \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}\|$. This last inequality and inequality (22) produce the Polyak-Łojasiewicz bound

$$
\frac{1}{2}\|\nabla h_\rho(\boldsymbol{x})\|^2 \;\geq\; \frac{\mu^2}{2(L + \rho)}[h_\rho(\boldsymbol{x}) - h_\rho(\boldsymbol{y})].
$$

Taking $\boldsymbol{x} = \boldsymbol{x}_k - \frac{1}{L+\rho}\nabla g_\rho(\boldsymbol{x}_k \mid \boldsymbol{x}_k) = \boldsymbol{x}_k - \frac{1}{L+\rho}\nabla h_\rho(\boldsymbol{x}_k)$, the Polyak-Łojasiewicz bound gives

$$
\begin{aligned}
h_\rho(\boldsymbol{x}_{k+1}) - h_\rho(\boldsymbol{x}_k) \;&\leq\; g_\rho(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k) - g_\rho(\boldsymbol{x}_k \mid \boldsymbol{x}_k) \\
&\leq\; g_\rho(\boldsymbol{x} \mid \boldsymbol{x}_k) - g_\rho(\boldsymbol{x}_k \mid \boldsymbol{x}_k) \\
&\leq\; -\frac{1}{L + \rho}\nabla g_\rho(\boldsymbol{x}_k \mid \boldsymbol{x}_k)^t\nabla h_\rho(\boldsymbol{x}_k) + \frac{1}{2(L + \rho)}\|\nabla h_\rho(\boldsymbol{x}_k)\|^2 \\
&=\; -\frac{1}{2(L + \rho)}\|\nabla h_\rho(\boldsymbol{x}_k)\|^2 \\
&\leq\; -\frac{\mu^2}{2(L + \rho)^2}[h_\rho(\boldsymbol{x}_k) - h_\rho(\boldsymbol{y})].
\end{aligned}
$$

Rearranging this inequality yields

$$h_\rho(\boldsymbol{x}_{k+1}) - h_\rho(\boldsymbol{y}) \leq \left[1 - \frac{\mu^2}{2(L+\rho)^2}\right][h_\rho(\boldsymbol{x}_k) - h_\rho(\boldsymbol{y})],$$

which can be iterated to give the stated bound. ∎

A.3. Proposition 3

Consider first the proximal distance iterates. The inequality

$$f(\boldsymbol{x}_k) + \frac{\rho_k}{2}\operatorname{dist}(\boldsymbol{x}_k, S)^2 \leq f(\boldsymbol{x}_k) + \frac{\rho_k}{2}\|\boldsymbol{x}_k - P_S(\boldsymbol{x}_{k-1})\|^2 \leq f[P_S(\boldsymbol{x}_{k-1})] \leq \sup_{\boldsymbol{x} \in S} f(\boldsymbol{x})$$

plus the coerciveness of $f(\boldsymbol{x})$ imply that $\boldsymbol{x}_k$ is a bounded sequence. The claimed bound now holds for c equal to the finite supremum of the sequence $2[\sup_{\boldsymbol{x} \in S} f(\boldsymbol{x}) - f(\boldsymbol{x}_k)]$. If in addition, $f(\boldsymbol{x})$ is continuously differentiable, then the stationarity equation

$$\boldsymbol{0} = \nabla f(\boldsymbol{x}_k) + \rho_k[\boldsymbol{x}_k - P_S(\boldsymbol{x}_{k-1})]$$

and the Cauchy-Schwarz inequality give

$$\rho_k\|\boldsymbol{x}_k - P_S(\boldsymbol{x}_{k-1})\|^2 = -\nabla f(\boldsymbol{x}_k)^t[\boldsymbol{x}_k - P_S(\boldsymbol{x}_{k-1})] \leq \|\nabla f(\boldsymbol{x}_k)\| \cdot \|\boldsymbol{x}_k - P_S(\boldsymbol{x}_{k-1})\|.$$

Dividing this by $\|\boldsymbol{x}_k - P_S(\boldsymbol{x}_{k-1})\|$ and squaring further yield

$$\rho_k^2\operatorname{dist}(\boldsymbol{x}_k, S)^2 \leq \rho_k^2\|\boldsymbol{x}_k - P_S(\boldsymbol{x}_{k-1})\|^2 \leq \|\nabla f(\boldsymbol{x}_k)\|^2.$$

Taking $d = \sup_k \|\nabla f(\boldsymbol{x}_k)\|^2$ over the bounded sequence $\boldsymbol{x}_k$ completes the proof.

For the penalty method iterates, the bound

$$f(\boldsymbol{y}_k) + \frac{\rho_k}{2}\operatorname{dist}(\boldsymbol{y}_k, S)^2 \leq f(\boldsymbol{y})$$

is valid by definition, where $\boldsymbol{y}$ attains the constrained minimum. Thus, coerciveness implies that the sequence $\boldsymbol{y}_k$ is bounded. When $f(\boldsymbol{x})$ is continuously differentiable, the proof of the second claim also applies if we substitute $\boldsymbol{y}_k$ for $\boldsymbol{x}_k$ and $P_S(\boldsymbol{y}_k)$ for $P_S(\boldsymbol{x}_{k-1})$. ∎

A.4. Proposition 4

Because the function $f(\boldsymbol{x}) + \frac{\rho_k}{2}\operatorname{dist}(\boldsymbol{x}, S)^2$ is convex and has the value $f(\boldsymbol{y})$ and gradient $\nabla f(\boldsymbol{y})$ at a constrained minimum $\boldsymbol{y}$, the supporting hyperplane principle says

$$\begin{aligned} f(\boldsymbol{y}_k) + \frac{\rho_k}{2}\operatorname{dist}(\boldsymbol{y}_k, S)^2 &\geq f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^t(\boldsymbol{y}_k - \boldsymbol{y}) \\ &= f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^t[P_S(\boldsymbol{y}_k) - \boldsymbol{y}] + \nabla f(\boldsymbol{y})^t[\boldsymbol{y}_k - P_S(\boldsymbol{y}_k)]. \end{aligned}$$

The first-order optimality condition $\nabla f(\boldsymbol{y})^t[P_S(\boldsymbol{y}_k) - \boldsymbol{y}] \geq 0$ holds given $\boldsymbol{y}$ is a constrained minimum. Hence, the Cauchy-Schwarz inequality and Proposition 3 imply

$$\begin{aligned} f(\boldsymbol{y}) - f(\boldsymbol{y}_k) &\leq \frac{\rho_k}{2}\operatorname{dist}(\boldsymbol{y}_k, S)^2 - \nabla f(\boldsymbol{y})^t[\boldsymbol{y}_k - P_S(\boldsymbol{y}_k)] \\ &\leq \frac{\rho_k}{2}\operatorname{dist}(\boldsymbol{y}_k, S)^2 + \|\nabla f(\boldsymbol{y})\| \cdot \operatorname{dist}(\boldsymbol{y}_k, S) \\ &\leq \frac{d + 2\sqrt{d}\|\nabla f(\boldsymbol{y})\|}{2\rho_k}. \end{aligned}$$

∎

A.5. Proposition 5

Let $\boldsymbol{x}_k$ converge to $\boldsymbol{x}_\infty$ and $\boldsymbol{y}_k \in P_S(\boldsymbol{x}_k)$ converge to $\boldsymbol{y}_\infty$. For an arbitrary $\boldsymbol{y} \in S$, taking limits in the inequality $\|\boldsymbol{x}_k - \boldsymbol{y}_k\| \le \|\boldsymbol{x}_k - \boldsymbol{y}\|$ yields $\|\boldsymbol{x}_\infty - \boldsymbol{y}_\infty\| \le \|\boldsymbol{x}_\infty - \boldsymbol{y}\|$; consequently, $\boldsymbol{y}_\infty \in P_S(\boldsymbol{x}_\infty)$. To prove the second assertion, take $\boldsymbol{y}_k \in P_S(\boldsymbol{x}_k)$ and observe that

$$
\begin{aligned}
\|\boldsymbol{y}_k\| &\le \|\boldsymbol{x}_k - \boldsymbol{y}_k\| + \|\boldsymbol{x}_k\| \\
&\le \|\boldsymbol{x}_k - \boldsymbol{y}_1\| + \|\boldsymbol{x}_k\| \\
&\le \|\boldsymbol{x}_k - \boldsymbol{x}_1\| + \|\boldsymbol{x}_1 - \boldsymbol{y}_1\| + \|\boldsymbol{x}_k\| \\
&\le \|\boldsymbol{x}_k\| + \|\boldsymbol{x}_1\| + \operatorname{dist}(\boldsymbol{x}_1, S) + \|\boldsymbol{x}_k\|,
\end{aligned}
$$

which is bounded above by the constant $\operatorname{dist}(\boldsymbol{x}_1, S) + 3\sup_{m \ge 1} \|\boldsymbol{x}_m\|$. ∎

A.6. Proposition 6

In fact, a much stronger result holds. Since the function $s(\boldsymbol{x})$ of equation (8) is convex and finite, Alexandrov's theorem (Niculescu and Persson, 2006) implies that it is almost everywhere twice differentiable. In view of the identities $\frac{1}{2}\operatorname{dist}(\boldsymbol{x}, S)^2 = \frac{1}{2}\|\boldsymbol{x}\|^2 - s(\boldsymbol{x})$ and $\boldsymbol{x} - P_S(\boldsymbol{x}) = \nabla \frac{1}{2}\operatorname{dist}(\boldsymbol{x}, S)^2$ where $P_S(\boldsymbol{x})$ is single valued, it follows that $P_S(\boldsymbol{x}) = \nabla s(\boldsymbol{x})$ is almost everywhere differentiable. ∎

A.7. Proposition 8

See the discussion just prior to the statement of the proposition. ∎

A.8. Proposition 9

The strong-convexity inequality

$$
g_\rho(\boldsymbol{x}_k \mid \boldsymbol{x}_k) \ge g_\rho(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k) + \frac{\mu}{2}\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\|^2
$$

and the tangency and domination properties of the algorithm imply

$$
h_\rho(\boldsymbol{x}_k) - h_\rho(\boldsymbol{x}_{k+1}) \ge \frac{\mu}{2}\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\|^2. \tag{23}
$$

Since the difference in function values tends to 0, this validates the stated limit. The remaining assertions follow from Propositions 7.3.3 and 7.3.5 of (Lange, 2016). ∎

A.9. Proposition 10

Let the subsequence x_{k_m} of the MM sequence $\boldsymbol{x}_{k+1} \in M(\boldsymbol{x}_k)$ converge to $\boldsymbol{z}$. By passing to a subsubsequence if necessary, we may suppose that $\boldsymbol{x}_{k_m+1}$ converges to $\boldsymbol{y}$. Owing to our closedness assumption, $\boldsymbol{y} \in M(\boldsymbol{z})$. Given that $h(\boldsymbol{y}) = h(\boldsymbol{z})$, it is obvious that $\boldsymbol{z}$ also minimizes $g(\boldsymbol{x} \mid \boldsymbol{z})$ and that $\boldsymbol{0} = \nabla g(\boldsymbol{z} \mid \boldsymbol{z})$. Since the difference $\Delta(\boldsymbol{x} \mid \boldsymbol{z}) = g(\boldsymbol{x} \mid \boldsymbol{z}) - h(\boldsymbol{x})$ achieves its minimum at $\boldsymbol{x} = \boldsymbol{z}$, the Fréchet subdifferential $\partial^F \Delta(\boldsymbol{x} \mid \boldsymbol{z})$ satisfies

$$
\boldsymbol{0} \in \partial^F \Delta(\boldsymbol{z} \mid \boldsymbol{z}) = \nabla g(\boldsymbol{z} \mid \boldsymbol{z}) + \partial^F(-h)(\boldsymbol{z}).
$$

It follows that $\boldsymbol{0} \in \partial^F(-h)(\boldsymbol{z})$. ∎

A.10. Proposition 11

Because $\Delta(\boldsymbol{x} \mid \boldsymbol{y}) = g(\boldsymbol{x} \mid \boldsymbol{y}) - h(\boldsymbol{x})$ achieves its minimum at $\boldsymbol{x} = \boldsymbol{y}$, the Fréchet subdifferential $\partial^F \Delta(\boldsymbol{x} \mid \boldsymbol{y})$ satisfies

$$\boldsymbol{0} \in \partial^F \Delta(\boldsymbol{y} \mid \boldsymbol{y}) = \nabla g(\boldsymbol{y} \mid \boldsymbol{y}) + \partial^F(-h)(\boldsymbol{y}).$$

It follows that $-\nabla g(\boldsymbol{y} \mid \boldsymbol{y}) \in \partial^F(-h)(\boldsymbol{y})$. Furthermore, by assumption

$$\|\nabla g(\boldsymbol{a} \mid \boldsymbol{x}_k) - \nabla g(\boldsymbol{b} \mid \boldsymbol{x}_k)\| \leq L\|\boldsymbol{a} - \boldsymbol{b}\|$$

for all relevant $\boldsymbol{a}$ and $\boldsymbol{b}$ and $\boldsymbol{x}_k$. In particular, because $\nabla g(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k) = \boldsymbol{0}$, we have

$$\|\nabla g(\boldsymbol{x}_k \mid \boldsymbol{x}_k)\| \leq L\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|. \tag{24}$$

Let W denote the set of limit points. The objective $h(\boldsymbol{x})$ is constant on W with value $\bar{h} = \lim_{k \to \infty} h(\boldsymbol{x}_k)$. According to the Łojasiewicz inequality applied for the subanalytic function $\bar{h} - h(\boldsymbol{x})$, for each $\boldsymbol{z} \in W$ there exists an open ball $B_{r(\boldsymbol{z})}(\boldsymbol{z})$ of radius $r(\boldsymbol{z})$ around $\boldsymbol{z}$ and an exponent $\theta(\boldsymbol{z}) \in [0, 1)$ such that

$$|h(\boldsymbol{u}) - h(\boldsymbol{z})|^{\theta(\boldsymbol{z})} = |\bar{h} - h(\boldsymbol{u}) - \bar{h} + \bar{h}|^{\theta(\boldsymbol{z})} \leq c(\boldsymbol{z})\|\boldsymbol{v}\|$$

for all $\boldsymbol{u} \in B_{r(\boldsymbol{z})}(\boldsymbol{z})$ and all $\boldsymbol{v} \in \partial^F(\bar{h} - h)(\boldsymbol{u}) = \partial^F(-h)(\boldsymbol{u})$. We will apply this inequality to $\boldsymbol{u} = \boldsymbol{x}_k$ and $\boldsymbol{v} = -\nabla g(\boldsymbol{x}_k \mid \boldsymbol{x}_k)$. In so doing, we would like to assume that the exponent $\theta(\boldsymbol{z})$ and constant $c(\boldsymbol{z})$ do not depend on $\boldsymbol{z}$. With this end in mind, cover the compact set W by a finite number of balls $B_{r(\boldsymbol{z}_i)}(\boldsymbol{z}_i)$ and take $\theta = \max_i \theta(\boldsymbol{z}_i) < 1$ and $c = \max_i c(\boldsymbol{z}_i)$. For a sufficiently large K, every $\boldsymbol{x}_k$ with $k \geq K$ falls within one of these balls and satisfies $|\bar{h} - h(\boldsymbol{x}_k)| < 1$. Without loss of generality assume $K = 0$. The Łojasiewicz inequality reads

$$|\bar{h} - h(\boldsymbol{x}_k)|^{\theta} \leq c\|\nabla g(\boldsymbol{x}_k \mid \boldsymbol{x}_k)\|. \tag{25}$$

In combination with the concavity of the function $t^{1-\theta}$ on $[0, \infty)$, inequalities (23), (24), and (25) imply

$$
\begin{aligned}
[h(\boldsymbol{x}_k) - \bar{h}]^{1-\theta} - [h(\boldsymbol{x}_{k+1}) - \bar{h}]^{1-\theta} &\geq \frac{1-\theta}{[h(\boldsymbol{x}_k) - \bar{h}]^{\theta}}[h(\boldsymbol{x}_k) - h(\boldsymbol{x}_{k+1})] \\
&\geq \frac{1-\theta}{c\|\nabla g(\boldsymbol{x}_k \mid \boldsymbol{x}_k)\|}\frac{\mu}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2 \\
&\geq \frac{(1-\theta)\mu}{2cL}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|.
\end{aligned}
$$

Rearranging this inequality and summing over k yield

$$\sum_{n=0}^{\infty} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| \leq \frac{2cL}{(1-\theta)\mu}[h(\boldsymbol{x}_0) - \bar{h}]^{1-\theta}$$

Thus, the sequence $\boldsymbol{x}_k$ is a fast Cauchy sequence and converges to a unique limit in W. ∎

References

Erik Agrell, Thomas Eriksson, Alexander Vardy, and Kenneth Zeger. Closest point search in lattices. *IEEE Transactions on Information Theory*, 48(8):2201–2214, 2002.

Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2003.

Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

Heinz H Bauschke and Patrick L Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Edward J Beltrami. *An Algorithmic Approach to Nonlinear Analysis and Optimization*. Academic Press, 1970.

Alberto Bemporad. A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares. *IEEE Transactions on Automatic Control*, 63(2):525–531, 2018.

Abraham Berman and Robert J Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Classics in Applied Mathematics. SIAM, 1994.

Quentin Berthet and Philippe Rigollet. Complexity theoretic lower bounds for sparse principal component detection. In *Conference on Learning Theory*, pages 1046–1066, 2013a.

Quentin Berthet and Philippe Rigollet. Optimal detection of sparse principal components in high dimension. *The Annals of Statistics*, 41(4):1780–1815, 2013b.

Edward Bierstone and Pierre D Milman. Semianalytic and subanalytic sets. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 67(1):5–42, 1988.

Jacek Bochnak, Michel Coste, and Marie-Françoise Roy. *Real Algebraic Geometry*, volume 36. Springer Science & Business Media, 2013.

Jérôme Bolte, Aris Daniilidis, and Adrian Lewis. The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4):1205–1223, 2007.

Jonathan M Borwein and Adrian S Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. CMS Books in Mathematics. Springer, New York, 2nd edition, 2006.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009.

James P Boyle and Richard L Dykstra. A method for finding projections onto the intersection of convex sets in Hilbert spaces. In *Advances in Order Restricted Statistical Inference*, pages 28–47. Springer, 1986.

Florentina Bunea, Alexandre B Tsybakov, Marten H Wegkamp, and Adrian Barbu. Spades and mixture models. *The Annals of Statistics*, 38(4):2525–2558, 2010.

Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20:1956–1982, 2010.

Emmanuel J Candès and Terence Tao. The power of convex relaxation: near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56:2053–2080, 2010.

Caihua Chen, Bingsheng He, and Xiaoming Yuan. Matrix completion via an alternating direction method. *IMA Journal of Numerical Analysis*, 32:227–245, 2012.

Eric C Chi, Hua Zhou, and Kenneth Lange. Distance majorization and its applications. *Mathematical Programming Series A*, 146:409–436, 2014.

Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer, 2011.

Richard Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49:1–23, 1943.

Ying Cui, Jong-Shi Pang, and Bodhisattva Sen. Composite difference-max programs for modern statistical estimation problems. *SIAM Journal on Optimization*, 28(4):3344–3374, 2018.

Alexandre D'Aspremont, Laurent El Ghaoui, Michael I Jordan, and Gert R G Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3): 434–448, 2007.

Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

Ky Fan. On a theorem of Weyl concerning eigenvalues of linear transformations I. *Proceedings of the National Academy of Sciences of the United States of America*, 35:652–655, 1949.

David Chin-Lung Fong and Michael Saunders. Lsmr: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441, 2008. ISSN 1468-4357.

Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1):59–99, 2015.

Gene H Golub and Charles F Van Loan. *Matrix Computations*. JHU Press, 3rd edition, 2012.

Marshall Hall and Morris Newman. Copositive and completely positive quadratic forms. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 59, pages 329–339. Cambridge Univ Press, 1963.

Rob Heylen, Dževdet Burazerović, and Paul Scheunders. Fully constrained least squares spectral unmixing by simplex projection. *IEEE Transactions on Geoscience and Remote Sensing*, 49(11):4112–4122, Nov 2011.

Nicholas J Higham. Computing the nearest correlation matrix - a problem from finance. *IMA Journal of Numerical Analysis*, 22(3):329–343, 2002.

Jean-Baptiste Hiriart-Urruty and Alberto Seeger. A variational approach to copositive matrices. *SIAM Review*, 52:593–629, 2010.

Harold Hotelling. Analysis of a complex of statistical variables into principle components. *Journal of Educational Psychology*, 24:417–441, 1933.

David R Hunter and Kenneth Lange. A tutorial on MM algorithms. *American Statistician*, 58:30–37, 2004.

Charles R Johnson and Robert Reams. Constructing copositive matrices from interior matrices. *Electronic Journal of Linear Algebra*, 17:9–20, 2008.

Iain M Johnstone and Arthur Yu Lu. On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486): 682–693, 2009.

Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11:517–553, 2010.

Yangyang Kang, Zhihua Zhang, and Wu-Jun Li. On the global convergence of majorization minimization algorithms for nonconvex optimization problems. *arXiv preprint arXiv:1504.07791*, 2015.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

Nirmal Keshava. A survey of spectral unmixing algorithms. *Lincoln Laboratory Journal*, 14 (1):55–78, 2003.

Henk AL Kiers. Majorization as a tool for optimizing a class of matrix functions. *Psychometrika*, 55:417–428, 1990.

Henk AL Kiers and Jos MF ten Berge. Minimization of a class of matrix trace functions by means of refined majorization. *Psychometrika*, 57:371–382, 1992.

Mark Aleksandrovich Krasnosel'skii. Two remarks on the method of successive approximations. *Uspekhi Matematicheskikh Nauk*, 10(1):123–127, 1955.

Alexander Ya Kruger. On Fréchet subdifferentials. *Journal of Mathematical Sciences*, 116 (3):3325–3358, 2003.

Louis Landweber. An iteration formula for Fredholm integral equations of the first kind. *American Journal of Mathematics*, 73(3):615–624, 1951.

Kenneth Lange. *Optimization.* Springer, New York, 2nd edition, 2010.

Kenneth Lange. *MM Optimization Algorithms.* SIAM, 2016.

Kenneth Lange and Kevin L Keys. The proximal distance algorithm. In *Proceedings of the 2014 International Congress of Mathematicians*, pages 95–116. Kyung Moon, August 2014.

Hoai An Le Thi, Van Ngai Huynh, and Tao Pham Dinh. Convergence analysis of DC algorithm for DC programming with subanalytic data. *Annals of Operation Research Technical Report, LMI, INSA-Rouen*, 2009.

Huan Li, Cong Fang, and Zhouchen Lin. Convergence rates analysis of the quadratic penalty method and its applications to decentralized distributed optimization. *arXiv preprint arXiv:1711.10802*, 2017.

Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

Miles Lubin and Iain Dunning. Computing in operations research using Julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

David G Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*, volume 2. Springer, 1984.

Julien Mairal. Optimization with first-order surrogate functions. In *International Conference on Machine Learning*, pages 783–791, 2013.

W Robert Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4(3):506–510, 1953.

Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11: 2287–2322, 2010.

Jean-Jacques Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes Rendus de l'Académie des Sciences de Paris A*, 255:2897–2899, 1962.

Katta G Murty and Feng-Tien Yu. *Linear Complementarity, Linear and Nonlinear Programming.* Heldermann Verlag, West Berlin, 1988.

Constantin Niculescu and Lars-Erik Persson. *Convex Functions and their Applications: A Contemporary Approach.* Springer, 2006.

Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journel of Optimization Theory and Applications*, pages 1–27, 2016.

Christopher C Paige and Michael A Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8 (1):43–71, 1982a.

Christopher C Paige and Michael A Saunders. Algorithm 583: LSQR: Sparse linear equations and least squares problems. *ACM Transactions on Mathematical Software (TOMS)*, 8(2): 195–209, 1982b.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

Boris T Polyak. The convergence rate of the penalty function method. *USSR Computational Mathematics and Mathematical Physics*, 11(1):1–12, 1971.

Haipeng Shen and Jianhua Z Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99:1015–1034, 2008.

Weijie Su, Stephen Boyd, and Emmanuel Candès. A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

Andreas Wächter and Lorenz T Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

Daniela M Witten, Robert Tibshirani, and Trevor Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

Jason Xu, Eric Chi, and Kenneth Lange. Generalized linear model regression under distance-to-set penalties. In *Advances in Neural Information Processing Systems*, pages 1385–1394, 2017.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal components analysis. *Journal of Computational and Graphical Statistics*, 15(2):262–282, 2006.