

Robust Principal Component Analysis

by

Arshak Minasyan

Bachelor of Science, Higher School of Economics, 2016

A thesis submitted in partial satisfaction of

the requirements for the degree of

Master of Science

in

Computer & Information Science

in the

COLLEGE OF SCIENCE AND ENGINEERING

of the

AMERICAN UNIVERSITY OF ARMENIA

Supervisor: _____

Signature: _____ Date: _____

Committee Member: _____

Signature: _____ Date: _____

Committee Member: _____

Signature: _____ Date: _____

Committee Member: _____

Signature: _____ Date: _____

Abstract

Robust Principal Component Analysis

Author: Arshak Minasyan

One of the most famous dimensionality reduction methods is Principal Component Analysis (PCA), which is successfully used worldwide. However this method is sensitive to outliers and hence a few number of them cause bias in the resulting subspace. There are a number of techniques now for the robustification of PCA, but we stick to the version introduced in [30]. The numerical technique for optimization in [30] relied on Iteratively Reweighted Least Squares (IRLS) method. In the present paper we adopted the Conjugate Gradient Descent algorithm with orthogonal matrix constraints from [18] for solving the non-convex matrix optimization problem. We discuss the arising computational and convergence problems and compare effectiveness of the methods.

Keywords: Robustness, Principal component analysis, nonconvex optimization, Stiefel manifold, Iteratively reweighted least squares, Conjugate gradient, Orthogonal matrices.

for the OC itself. You may not charge a fee for the sole service of providing access to and/or use of the OC via a network (e.g. the Internet), whether it be via the world wide web, FTP, or any other method.

2. You may modify your copy or copies of the OpenContent or any portion of it, thus forming works based on the Content, and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified content to carry prominent notices stating that you changed it, the exact nature and content of the changes, and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the OC or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License, unless otherwise permitted under applicable Fair Use law.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the OC, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the OC, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Exceptions are made to this requirement to release modified works free of charge under this license only in compliance with Fair Use law where applicable.

3. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, distribute or modify the OC. These actions are prohibited by law if you do not accept this License. Therefore, by distributing or translating the OC, or by deriving works herefrom, you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or translating the OC.

NO WARRANTY

4. BECAUSE THE OPENCONTENT (OC) IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE OC, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE OC "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK OF USE OF THE OC IS WITH YOU. SHOULD THE OC PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE YOU ASSUME THE COST OF ALL NECESSARY REPAIR OR CORRECTION.

Licenses for Software and Content

Software Copyright License (to be distributed with software developed for masters project)

Copyright (c) 2018, Arshak Minasyan

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

(This license is known as "The MIT License" and can be found at <http://opensource.org/licenses/mit-license.php>)


Content Copyright License (to be included with Technical Report)

LICENSE

Terms and Conditions for Copying, Distributing, and Modifying

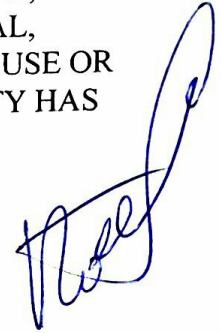
Items other than copying, distributing, and modifying the Content with which this license was distributed (such as using, etc.) are outside the scope of this license.

1. You may copy and distribute exact replicas of the OpenContent (OC) as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the OC a copy of this License along with the OC. You may at your option charge a fee for the media and/or handling involved in creating a unique copy of the OC for use offline, you may at your option offer instructional support for the OC in exchange for a fee, or you may at your option offer warranty in exchange for a fee. You may not charge

No Software


5. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MIRROR AND/OR REDISTRIBUTE THE OC AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE OC, EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

(This license is known as "OpenContent License (OPL)" and can be found at <http://opencontent.org/opl.shtml>)



Checklist for Submitting the Thesis Report and Accompanying Data

Submit the following:

1. One spiral bound hard copy of the Thesis Report; title pages should have the Committee signatures
2. CD-ROM containing the publication file of your thesis report (Word or PDF version), any relevant data used in the study, as well as any software that was written as part of your research
3. Signed copyright release form

Robust Principal Component Analysis

Arshak Minasyan

Received: date / Accepted: date

Abstract One of the most famous dimensionality reduction methods is Principal Component Analysis (PCA), which is successfully used worldwide. However this method is sensitive to outliers and hence a few number of them cause bias in the resulting subspace. There are a number of techniques now for the robustification of PCA, but we stick to the version introduced in [30]. The numerical technique for optimization in [30] relied on Iteratively Re-weighted Least Squares (IRLS) method. In the present paper we adopted the Conjugate Gradient Descent algorithm with orthogonal matrix constraints from [18] for solving the non-convex matrix optimization problem. We discuss the arising computational and convergence problems and compare effectiveness of the methods.

Keywords Robustness · Principal component analysis · Nonconvex optimization · Stiefel manifold · Iteratively reweighted least squares · Conjugate gradient · Orthogonal matrices

1 Introduction

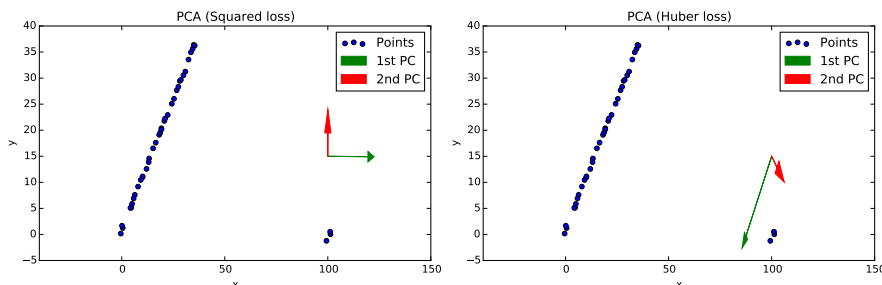
The general problem of data analysis in high dimensions is arising in many fields, such as computer vision [33], [26], signal processing in medicine [10], [11], etc. In some situations there is an underlying structure of high-dimensional data, i.e. there is a low-rank approximation $\mathcal{X}_{\text{low-rank}} \in \mathbf{R}^{N \times d}$ of initial data $\mathcal{X} \in \mathbf{R}^{N \times D}$, where d is the dimension of low-dimensional space with $d \ll D$. Here N is the number of observations and D — dimension of each observation, $D \gg 1$. The standard method of finding such structures is known as principal component analysis (PCA) and was pioneered by Pearson [29], see [22]. PCA is the simplest method of dimensionality reduction, since it only requires

computation of the first few singular values of the matrix \mathbf{X} . First d principal components form a d -dimensional subspace to where the data points are projected. Computing singular values can be done effectively as indicated in [31], [23], [20], [4]. LAPACK [3] provides a number of methods for computing singular values. The choice of d is discussed in review paper [1], in references therein and in recent paper [14].

It is a well-known fact that PCA is sensitive to outliers. The term "robustness" was first introduced by G. Box [6] in 1953. The monograph [21] of Paul Huber in 1981 had a great impact on modern statistics and developed robustness theory. The question of constructing a robust method of dimensionality reduction has been discussed recently, see [9], [38], [15], [24], [25], [36], [37]. In this work we discuss the same "robust" setup as described in [30] and relying on Huber's approach. This approach differs from the ones mentioned before and is based on constructing a matrix optimization problem the solution of which gives the "robust" low-dimensional space. The objective function is itself smooth and convex, however the constraints ($X^T X = I_p$) are non-convex.

Being sensitive to outliers in this setup means that significant principal components will be rotated provided even small number of outliers. For the simplicity of illustrations of this phenomenon we manually generate data points with underlying structure (a line segment) then add a few outliers laying far from it. We expect from the robust method to ignore the outliers and find the direction of the line as the first principal component's direction. The standard PCA would not get the direction right because of existing outliers.

The following figure illustrates the robustness of Huber function defined by (3). In the first plot one can see how the principal components are affected by outliers and the eigenvector's directions are rotated, while in the second plot with Huber loss function the first principal components have the correct direction figuring out the direction of the line and ignoring the outliers.



During last few decades there are many methods developed for handling the optimization problems with orthogonal matrix constraints. To name just a few of them see [18], [2], [19], [5], [35]. One of the important parts of this work are numerical experiments aimed to solve optimization problems with smooth convex functions and orthogonal constraints. The conjugate gradient method on Stiefel manifold adopted for [18] is taken as the basic one. The results were

compared with method of IRLS used to solve the problem of robust PCA in [30] for different datasets.

The structure of the paper is the following. Section 2 describes the "robust" approach based on Huber loss function. The optimization methods for solving the proposed problem from Section 2 are collected in Section 3. Section 4 contains a number of examples where these methods can be applied with careful analysis of their convergence, time complexity and accuracy. Section 5 sums up previous sections and contains several concluding remarks.

Notation. Throughout the paper the following notation is used. \mathbf{R} denotes the set of real numbers. For a vector $\mathbf{v} \in \mathbf{R}^n$ we denote ℓ_p norm: $\|\mathbf{v}\|_p := (\sum_{i=1}^n v_i^p)^{1/p}$, $p = 2$ is known as the Euclidean norm. Denote $\mathbb{M}_{n,m}$ as a set of all real matrices of size $n \times m$ and \mathcal{S}_n the set of all symmetric matrices of size $n \times n$. We use superscript \cdot^k for the optimizing variable to be a variable value at iteration k . The standard scalar product $\langle \cdot, \cdot \rangle_E$ with matrix entries is defined by $\langle A, B \rangle_E = \text{tr}(A^T B)$ which is also known as Frobenius inner product. As for the canonical metric on Stiefel manifold define $\langle A, B \rangle_S = \text{tr} A^T (I - \frac{1}{2} X X^T) B$, where X is the point on manifold $\text{St}(X) = \{X \in \mathbb{M}_{n,k} : X^T X = I_k\}$ at which the inner product is computed. The Frobenius norm of matrix $A \in \mathbb{M}_{m,n}$ is given by $\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2} = \sqrt{\text{tr} A^T A}$.

2 Robust Principal Component Analysis

The problem of constructing robust version for the PCA is studied a lot recently and a number of techniques are available [9], [38], [15], [24], [25], [36], [37]. However, in this work we mainly concentrate on the robust version of PCA first introduced by B. Polyak and M. Khlebnikov [30].

For getting started assume a cluster of points $\mathcal{X} = \{X_1, \dots, X_N\}$, where each observation $X_i \in \mathbf{R}^D$, where $D \gg 1$. The aim is to find the lower-dimensional structure of initial cluster \mathcal{X} . The standard PCA method computes the first d eigenvectors of the sample covariance matrix $\hat{\Sigma}$ defined as

$$\hat{\Sigma} = \frac{1}{N} \sum_{i=1}^N (X_i - \bar{X})(X_i - \bar{X})^T, \quad \bar{X} = \frac{1}{N} \sum_{i=1}^N X_i.$$

The first $d \ll D$ eigenvectors span the subspace to where the points X_1, \dots, X_N are projected to. Formally,

$$\hat{\Sigma} v_i = \lambda_i v_i, \quad i = 1, \dots, d.$$

and the lower dimensional subspace is $\Pi = \text{span}\{v_1, \dots, v_d\}$ and v_i are called the principal components.

Finding the principal components can be done by finding the solution of an optimization problem which aimed to minimize the sum of squared distances

between given cluster points \mathcal{X} and the hyperplane $Cx = b$, where $C \in \mathbf{R}^{d \times D}$ and $b \in \mathbf{R}$. Hence we want to find such b and C such that the

$$\sum_{i=1}^N \|CX_i - b\|_2^2$$

is the least possible. However, one can easily note that taking $b = 0$ and $C = 0$ (i.e., zero-vector(matrix) in their corresponding dimensions) leads to the sum of 0, which is the least possible but contains no information about the lower-dimensional structure of \mathcal{X} . This brings us to the following optimization problem

$$\min_{b, CC^T=I} \sum_{i=1}^N \|CX_i - b\|_2^2. \quad (1)$$

It is known (see e.g. [30]) that the minimum of problem (1) is achieved for

$$b^* = C^* \bar{X}, \quad C^* = \begin{pmatrix} v_{(D)}^T \\ v_{(D-1)}^T \\ \dots \\ v_{(D-d+1)}^T \end{pmatrix}, \quad (2)$$

where $v_{(i)}$ s are the normalized eigenvectors of matrix $\hat{\Sigma}$ associated with the i -th largest in absolute value eigenvalue, namely $v_{(D)} = v_{\min}$. This yields the hyperplane orthogonal to Π .

We note that the problem (1) is non-convex in general. However, due to ℓ_2 norm there is a closed-form solution of (1).

However, ℓ_2 norm has its drawbacks, namely the associated loss function is very sensitive to outliers. That is, it tries to fit all the points equivalently and a single outlier can possibly make the fitting model neglect the underlying structure of other points. This is the main issue with standard PCA.

The "robustified" version of optimization problem (1) can be obtained using the robustness of ℓ_1 norm and easiness of ℓ_2 norm expressed through the Huber function defined as

$$h(x) = \begin{cases} x^2/2, & \text{if } |x| < \delta \\ \delta|x| - \delta^2/2, & \text{if } |x| \geq \delta \end{cases} \quad (3)$$

and the corresponding optimization problem reads as follows

$$\min_{b, CC^T=I} \sum_{i=1}^N h(\|CX_i - b\|_2). \quad (4)$$

The brief description of algorithm of *iteratively reweighted least squares* which was used in [30] to solve (4) can be found below. The origin of this algorithm goes back to E. Weiszfeld [34] who first introduced this method for solving Fermat-Weber problem [7].

The initial step of the algorithm for solving (4) is the standard PCA step, i.e. taking

$$\bar{X}^0 = \frac{1}{N} \sum_{i=1}^N X_i, \quad \Sigma^0 = \frac{1}{N} \sum_{i=1}^N (X_i - \bar{X}^0)(X_i - \bar{X}^0)^T$$

yields the values of C_1 and b_1 as follows

$$C_1 = \begin{pmatrix} u_{(D)}^T \\ u_{(D-1)}^T \\ \dots \\ u_{(D-d+1)}^T \end{pmatrix}, \quad b_1 = C_1 \bar{X}^0, \quad (5)$$

where $u_{(i)}$ is the normalized eigenvector of Σ^0 associated with the i -th largest in absolute value eigenvalue.

In the k -th iteration take

$$w_{ik} = \min \left\{ 1, \frac{\delta}{\|C_k X_i - b_k\|_2} \right\}. \quad (6)$$

Then, update

$$\bar{X}^k = \frac{\sum_{i=1}^N X_i w_{ik}}{\sum_{i=1}^N w_{ik}}, \quad \Sigma^k = \frac{\sum_{i=1}^N w_{ik} (X_i - \bar{X}^k)(X_i - \bar{X}^k)^T}{\sum_{i=1}^N w_{ik}}$$

and compute the eigenvectors of matrix Σ^k for updating variables C_k and b_k according to (2). Repeat updating \bar{X}^k and Σ^k until convergence.

If function $h(x)$ differs from (3) but remains symmetric, differentiable and convex, weights w_{ik} are calculated as

$$w_{ik} = \frac{h'(\varepsilon_{ik})}{\varepsilon_{ik}}, \quad \varepsilon_{ik} = \|C_k X_i - b_k\|_2 \quad (7)$$

instead of (6).

Further details related IRLS algorithm can be found in [30].

3 Minimization on Stiefel manifold

This section contains an optimization algorithm adopted from [18]. It also contains preliminaries on Stiefel manifold as well as facts on which the conjugate gradient algorithm mainly relies.

The aim of this section is to describe a method for solving the optimization problem

$$\min_{X^T X = I} F(X), \quad (8)$$

where F is smooth and convex in $X \in \mathbb{M}_{n,p}$. We remind that $F(X)$ is differentiable means that $F(X + \Delta) = F(X) + \langle F_X, \Delta \rangle + o(\Delta)$ with standard Frobenius matrix product $\langle \cdot, \cdot \rangle$ and $F_X \in \mathbb{M}_{n,p}$.

Define the Stiefel manifold as follows $\mathbf{St}_{n,p} = \{X \in \mathbb{M}_{n,p} : X^T X = I_p\}$. The two important special cases of Stiefel manifold are sphere \mathcal{S}^{n-1} in \mathbf{R}^n when $p = 1$ and \mathcal{O}_n the group of all invertible matrices with unit determinant for the case of $p = n$. \mathcal{O}_n is also known as *orthogonal group*.

The canonical inner product of Stiefel manifold reads as

$$\langle A, B \rangle_S = \text{tr} A^T \left(I - \frac{1}{2} X X^T \right) B, \quad (9)$$

which is dependent on the point of manifold X where the scalar product is computed.

Theorem 2.1 from [18] provides a method of computing the geodesic equation in $O(np^2)$. The Corollary 2.2 contains rather closed form and friendly expressions for geodesic equation. We repeat this result below.

The geodesic equation for moving from $X(0) = X$ in the direction of $\dot{X}(0) = H$ on Stiefel manifold has the following form

$$X(t) = X M(t) + Q N(t), \quad (10)$$

where $QR = K := (I - X X^T)H$ is the compact QR-decomposition of K , $A = X^T H$ and

$$\begin{pmatrix} M(t) \\ N(t) \end{pmatrix} = \exp \left\{ t \begin{pmatrix} A & -R^T \\ R & 0 \end{pmatrix} \right\} \begin{pmatrix} I_p \\ 0 \end{pmatrix} \quad (11)$$

The gradient of the function $F(X)$ on the Stiefel manifold is defined to be the tangent vector at X , hence using the inner product (9) we get

$$\nabla F(X) := F_X - X F_X^T X. \quad (12)$$

These formulas are used in [18] to construct Steepest Descent method and Conjugate Gradient method for minimization problem (8). We slightly adopt the methods to deal with our problem (4), which can be written as

$$\min_{b, C C^T = I} F(b, C), \quad (13)$$

with $F(b, C) = \sum_{i=1}^N h(\|C X_i - b\|_2)$. For unconstrained minimization over b we apply steepest descent step at each iteration, while for minimization over C on Stiefel manifold we use conjugate gradient iterations. We denote G_k the gradient of $F(b_k, C_k)$ with respect to C on the manifold as in (12), and gradient with respect to b as g_k . With this notation in mind the algorithm for solving (13) reads as follows.

Algorithm 1 Conjugate Gradient on the Stiefel manifold

-
- 1: **Given:** problem $\min_{b, C} F(b, C)$ choose C_0 and some b_0 such that $C_0 C_0^T = I$.
 - 2: **Compute:** $G_0 = \nabla F(\cdot, C_0)$ and set $H_0 = -G_0$.
 - 3: **for** $k = 0, 1, \dots$ **do**
 - 4: **Minimize:** $F(b^k, C_k(t))$ over t where

$$C_k(t) = C^k M(t) + QN(t),$$

where $(I - (C^k)^T C^k)H_k = QR$ is a QR-decomposition. $M(t)$ and $N(t)$ are given in (11).

- 5: **Update:** $C^{k+1} = C_k(t_k)$ with $t_k = \arg \min_t F(b_k, C_k(t))$.
- 6: **Compute:** w_{ik} according to (6).
- 7: **Update:** $b^{k+1} = C^{k+1} \bar{X}_w$, where $\bar{X}_w := \frac{\sum_{i=1}^N X_i w_{ik}}{\sum_{i=1}^N w_{ik}}$.
- 8: **Compute:** $G_{k+1} = \nabla F(b^{k+1}, C^{k+1})$
- 9: **Parallel transport:**

$$\tau H_k = H_k M(t_k) - C_k R^T N(t_k), \quad (14)$$

$$\tau G_k = G_k \text{ (not parallel)} \quad (15)$$

- 10: **Update:** $H_{k+1} = -G_{k+1} + \gamma_k \tau H_k$, where

$$\gamma_k = \frac{\langle G_{k+1} - \tau G_k, G_{k+1} \rangle_S}{\langle G_k, G_k \rangle_S}$$

- 11: **Reset:** $H_{k+1} = -G_{k+1}$ if $k+1 \equiv 0 \pmod{d(D-d) + m(m-1)/2}$, where $m = D-d$.
 - 12: **end for**
-

There are no closed form expression for parallel translation for matrix G_k and hence we take it equal G_k . The choice $\tau G_k = 0$ works as well and there is no much difference in performance.

There are a lot more possible ways to choose γ_k , see page 17 of [18] and note that taking the coefficient γ_k being equal to 0 and skipping the parallel transport step transforms this method to the standard gradient descent algorithm on Stiefel manifold, i.e. at each step we get

$$H_{k+1} = -G_{k+1}.$$

Implementation details relate to 1D optimization problems (Steps 5), termination conditions (Step 11), calculation of QR-decomposition (Step 4); we do not discuss them here.

4 Numerical experiments

This section sums up the performance of described algorithm from the computational point of view comparing it with IRLS in terms of convergence, accuracy and time complexity.

The first examples treat standard eigenvalue problem (find eigenvectors of a symmetric matrix A corresponding to p smallest eigenvalues). The function

to be minimized is quadratic and vector b lacking. These results are added to check efficiency of the optimization method on Stiefel manifold compared with standard linear algebra techniques [31], [23], [20], [4], [3].

Implementation details. The code for the algorithms of conjugate gradient and IRLS were written in Python 2.7. Gradients were computed automatically using Theano¹ framework [27]. Our experiments were performed on a 64-bit, dual-core, Intel(R) 2.3GHz Xeon(R) CPU machine with 8 GB of memory, running Linux version 4.9.

4.1 Eigenvalue Problem

The simplest eigenvalue problem for finding eigenvector of $A \in S_n$ with the least eigenvalue reads

$$\min_{\|x\|_2=1} \frac{1}{2} x^T A x.$$

Considering the orthogonal matrix instead of unit-norm vector brings us to the following problem

$$\min_{X^T X = I_p} f(X) := \frac{1}{2} \text{tr} X^T A X, \quad (16)$$

where $X \in \mathbb{M}_{n,p}$, $p > 1$ and A is some symmetric matrix of size n . Its solution is provided by matrix X^* with p columns being eigenvectors with p smallest eigenvalues.

In figure 1 we would like to show for problem (16) the dependence between matrix size n and time and dimension p and time. We see that solution of problems with $n \leq 1000$ and $p \leq 50$ requires less than one minute of calculations. It is comparable with results for standard linear algebra tools. Notice that matrix A was not sparse, its entries were chosen to be i.i.d. standard normal random variables.

The rate of convergence of the method for the same problem (16) is given in figure 2.

The method demonstrated global convergence for all examples.

4.2 Weighted eigenvalue problem

A slight generalization of eigenvalue problem reads as

$$\min_{X^T X = I_p} \frac{1}{2} \text{tr} X^T A X \cdot N, \quad (17)$$

where $N \in \mathcal{S}_p(\mathbf{R})$ is a symmetric matrix of size p .

¹ <https://github.com/Theano/Theano>

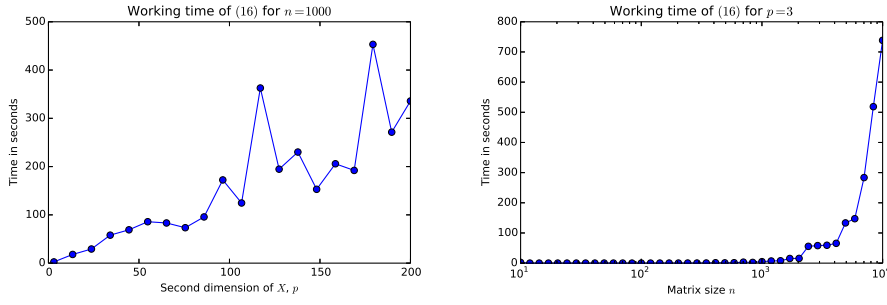


Fig. 1 This figure indicates the time needed for convergence of Algorithm 1 for two different situations. In the left plot we take a matrix of size $n = 1000$ and vary the dimension p of Stiefel manifold $\text{St}_{n,p}$. In the right plot the dependence for the fixed $p = 3$ on the matrix size is given.

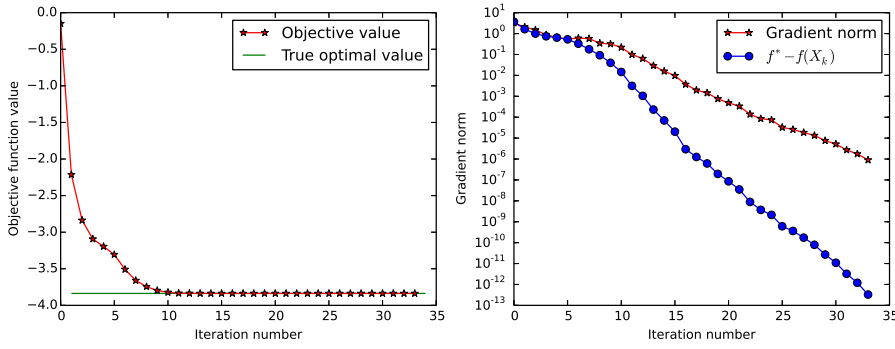


Fig. 2 Consider the problem (16) with $n = 20$ and $p = 3$. The plot in the left shows the objective function value in each iteration. In right plot we see two lines one of which is the Frobenius norm of gradient G_k and the second is the distance from the optimal value. These two lines illustrate the superlinear convergence of the method. The stopping criteria was the following: stop if $\|G_k\|_F < 10^{-6}$.

Here we have a simple explicit example borrowed from [5] such that reweighting of summands of type $x_i^T A x_i$ can really prevent global convergence. The example is for the manifold $\text{St}_{4,2}(n = 4, p = 2)$ and the matrices A and N are

$$A = \text{diag}(1, 2, 3, 4), \quad N = \text{diag}(1, 2).$$

First, let us note that for a diagonal matrix N with entries n_1, \dots, n_p (17) can be rewritten as

$$\min_{(x_i, x_j) = \delta_{ij}} \frac{1}{2} \sum_{i=1}^p n_i \cdot x_i^T A x_i,$$

where δ_{ij} is the Kronecker symbol. As for given example we end up with the following expression

$$\min_{\substack{(x_1, x_2) \neq 0, \\ \|x_1\| = \|x_2\| = 1}} \frac{1}{2} [x_1^T A x_1 + 2x_2^T A x_2]. \quad (18)$$

The eigenvalues of matrix A are $\lambda_{1,2,3,4} = 1, 2, 3, 4$ and it is easy to see that the global minimum is achieved for $x_1 = v_2$ (the eigenvector associated with second smallest eigenvalue) and $x_2 = v_1$ (the eigenvector associated with smallest eigenvalue) with function value of 2. So for this case the order of eigenvectors in matrix $X \in \mathbb{M}_{n,p}(\mathbf{R})$ is crucial.

Taking initial value X_0 as follows

$$X_0 = \begin{pmatrix} \frac{\sqrt{3}}{3} & -\frac{\sqrt{2}}{2} \\ 0 & 0 \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{3} & 0 \end{pmatrix}$$

makes the method of conjugate gradient to converge to $\hat{X} = (v_1 \ v_2)$ giving the objective function value of 2.5 with

$$\hat{X} = \begin{pmatrix} 9.9 \cdot 10^{-8} & -1. \\ 0. & 0. \\ -1. & -7.6 \cdot 10^{-8} \\ 8.8 \cdot 10^{-4} & 2.5 \cdot 10^{-5} \end{pmatrix}, \quad \hat{X}^T X = \begin{pmatrix} 1. & -2.1 \cdot 10^{-18} \\ -2.1 \cdot 10^{-18} & 1. \end{pmatrix}.$$

The function value along with its gradient norm are given in the figure 3

We note that in this subsection we skip the step 5 from algorithm 1 since there is no b in the optimization problems formulated for finding the eigenvectors of matrix A .

4.3 Robust PCA

Recall the problem of robust PCA from section 2 for given data points $\mathcal{X} = \{X_1, \dots, X_N\}$ and d – the number of rows of matrix size $C \in \mathbb{M}_{d,D}$:

$$\min_{b, C \in \mathbb{M}_{d,D}} \sum_{i=1}^N h(\|C X_i - b\|_2). \quad (19)$$

Further we will apply the described algorithm 1 for two fairly popular datasets.

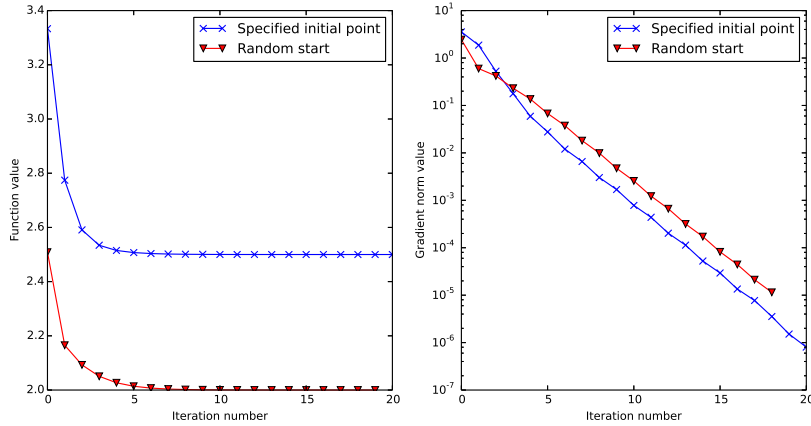


Fig. 3 The convergence of the method and optimality condition for problem (18) in two cases: specifically chosen initial point (only local convergence was observed) and random initial point (global convergence takes place).

4.3.1 Sleep in Mammals

We take the following [data](#)² and illustrate performances of discussed methods. From the whole pool of data that could be found in provided link we restrict ourselves to only 4 features of mammals. Those are: 'BodyWt' (body weight), 'BrainWt' (brain weight), 'LifeSpan' and 'Gestation'. Omitting the observations containing N/As we get the final data with $N = 55$ observations of $D = 4$ features. Prior to applying these methods we standardized the dataset, i.e. scaled it to the common span $0 \div 100$.

Our aim at this point is to construct the lower-dimensional subspace of $\mathbf{X} \in \mathbb{R}^{N \times D}$. Here, we have chosen $d = 2 \implies D - d = 2$.

We applied conjugate gradient method on Stiefel manifold as well as iteratively re-weighted least squares. For conjugate gradient we set the initial point to be equal to C_{PCA} and b_{PCA} just it was done in the method of IRLS. This boosts the convergence a lot.

The obtained values of these two algorithms are given below

$$C_{\text{CG}} = \begin{pmatrix} -0.65146285 & -0.74451327 & 0.09132868 & 0.113821 \\ -0.75610812 & 0.65134562 & -0.06097513 & -0.018205 \end{pmatrix}, \quad b_{\text{CG}} = \begin{pmatrix} 1.35270751 \\ 0.39414133 \end{pmatrix}$$

$$C_{\text{IRLS}} = \begin{pmatrix} -0.66576769 & 0.68983301 & -0.25604461 & 0.12379399 \\ 0.06612924 & 0.37813871 & 0.4596208 & -0.80086626 \end{pmatrix}, \quad b_{\text{IRLS}} = \begin{pmatrix} 1.35270751 \\ 0.39414133 \end{pmatrix}$$

² Sleep in Mammals: <http://www.statsci.org/data/general/sleep.html>

Method	Time [sec]	# Iterations	Value converged	Relative error	Gradient norm
IRLS	0.9	12	262.65993	$1.2 \cdot 10^{-11}$	–
CG on St	1.8	176	262.65993	$4.8 \cdot 10^{-09}$	$4.5 \cdot 10^{-5}$

Table 1 This table contains computational details of discussed methods for Sleep in Mammals dataset.

with the objective function value being approximately $\hat{f} = 262.659933$ for both IRLS and CG. The value of \hat{f} was rounded with 6 decimal digits precision, and moreover the relative error

$$\frac{\hat{f}_{\text{IRLS}} - \hat{f}_{\text{CG}}}{\hat{f}_{\text{IRLS}}} < 10^{-14}.$$

Moreover, matrices C_{CG}^T and C_{IRLS}^T indeed belong to $\mathbf{St}_{4,2}$:

$$C_{\text{CG}} C_{\text{CG}}^T = \begin{pmatrix} 1. & -1.3 \cdot 10^{-16} \\ -1.3 \cdot 10^{-16} & 1. \end{pmatrix}, \quad C_{\text{IRLS}} C_{\text{IRLS}}^T = \begin{pmatrix} 1. & -1.1 \cdot 10^{-16} \\ -1.1 \cdot 10^{-16} & 1. \end{pmatrix}$$

In addition, we provide optimal values of C and b for the problem (1) which.

$$C_{\text{PCA}} = \begin{pmatrix} -0.31910718 & -0.42073214 & -0.43450504 & 0.72963035 \\ -0.72146391 & 0.67672711 & -0.14620879 & -0.01237876 \end{pmatrix}, \quad b_{\text{PCA}} = \begin{pmatrix} 4.25917183 \\ -1.637549253 \end{pmatrix}$$

The discrepancy plot for above mentioned value of matrix C and b illustrates the robustness of the optimal values of problem (4) compared to the standard PCA.

The

4.3.2 Wine Quality

The other choice of dataset is the data of *wine quality*³. The detailed description of the dataset could be found in [13]. We took $D = 10$ features of red wine with $N = 1599$ and the same 10 features for white wine (with $N = 4898$). We consider these datasets separately and show that illustrated algorithms work in a reasonable amount of time. Further comparison of working time will be discussed later.

The features used are 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'density', 'pH', 'sulphates' and 'alcohol'. Again, we have chosen $d = 2$ and hence $D - d = 8$. In the table 4.3.2 we provide the working time and number of iterations for each algorithm. The optimal values are intentionally removed from the paper. However, the optimal values achieved by the algorithms and their relative errors are provided in the table 4.3.2.

³ <http://www3.dsi.uminho.pt/pcortez/wine/>

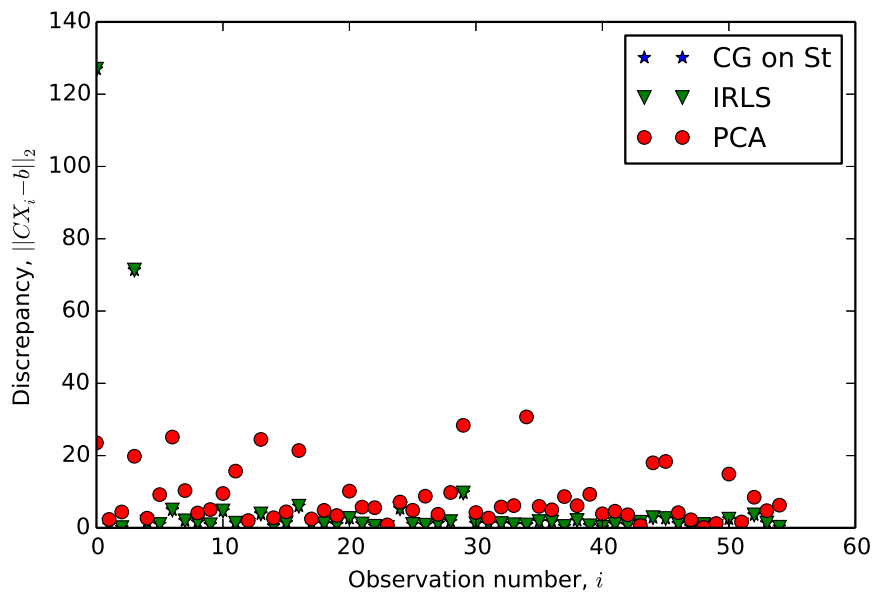


Fig. 4 Discrepancy plot of Sleep in Mammals dataset. Closer look shows that the hyperplane obtained by CG and IRLS have fairly high discrepancy for two points (Indian and African elephants) compared to others. Nevertheless the discrepancies of rest of points are very low. This means that the it actually finds the lower dimensional structure of the data. As for standard PCA we see that discrepancies have no such high values as in CG and IRLS cases, however the values are higher than that of CG and IRLS. The latter means that PCA is affected by the outliers.

Method	White Wine		Red Wine	
	Time [sec]	# Iterations	Time [sec]	# Iterations
IRLS	6.25	12	2.12	11
CG on St	19.12	186	5.64	119

Table 2 This table shows the time and number of iterations for two types of wine: white and red.

Method	White Wine		Red Wine	
	Value converged	Relative error	Value converged	Relative error
IRLS	3704.9952	$5.2 \cdot 10^{-9}$	1338.0631	$1.1 \cdot 10^{-9}$
CG on St	3704.9981	$2.5 \cdot 10^{-7}$	1338.5662	$7.5 \cdot 10^{-7}$

Table 3 This table shows the converged values as well as the relative error for two types of wine: white and red.

4.3.3 Discussion

For the eigenvalue problem we know that the extrema points of problem (16) are the collection of p eigenvectors and any collection besides the collection of p eigenvectors associated with smallest eigenvalues are stationary points. The theoretical validation of global convergence to the minimum point remains open, however, our numerical experience with random initial points confirmed global convergence. This method handles problems with large dimensions of matrix A in a reasonable amount of time. The situation for weighted eigenvalue problem (subsection 4.2) can be different; no global convergence can be guaranteed.

It is well known that conjugate gradient method for unconstrained minimization of $f(x)$ is finite (for $f(x)$ quadratic) or converges superlinearly (for $f(x)$ smooth and strongly convex [28]). We are not aware on any results on convergence and its rate for optimization on Stiefel manifolds. The practical implementation exhibits fast convergence for most examples. The algorithm IRLS shows a bit faster convergence however its iterations are costlier. For the large dataset sizes N and D used QR decomposition is easier to compute rather than SVD used in IRLS algorithm.

From the optimization point of view indeed many iterations of convergence are required to achieve high accuracy. However, in most machine learning tasks full optimization might give very little or no gain ultimately. Taking the initial point to be C_{PCA} and b_{PCA} and making only small number of iterations will be enough to handle the outliers and obtain more "robust" lower-dimensional hyperplane.

5 Concluding remarks

We have proposed an optimization method that deals with problem of the following type: $\min_{b, C C^T = I} F(b, C)$ for smooth and convex function $F(\cdot)$. The important special cases of such problems are the robust principal component analysis and (weighted) eigenvalue problem. The standard PCA is used in many fields of research but unfortunately is not robust to outliers. The presented robust version of PCA (4) can be solved using either IRLS method from [30] or described in this paper algorithm 1. Solving (4) indeed requires more computational resources than PCA. IRLS iterations are based on singular value decomposition [17] while CG on Stiefel manifold computes the QR decomposition in each iteration. However, there might be problems with time and memory while computing the SVD decomposition of given matrix $\mathcal{X} \in \mathbf{R}^{N \times D}$ for large enough N and D . While SVD (even with no full matrices) can be very time and space consuming the QR decomposition is less costly. Therefore, when the sizes N and D are large enough then it would be much better to use CG algorithm 1 rather than IRLS. There are cases when IRLS fails but CG converges in a reasonable amount of time.

References

1. H. Abdi, L. J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, **2**, 433–459, 2010.
2. P. Absil, R. Mahony, R. Sepulchre. Optimization algorithms on matrix manifolds. *Princeton Univ. Press*, 2008.
3. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen. LAPACK Users' Guide: Third Edition. *Society for Industrial and Applied Mathematics*, 1999.
4. M. Andrecut. Parallel GPU Implementation of Iterative PCA Algorithms. *Journal of Computational Biology*. **16** (11): 1593–1599, 2009.
5. P. Birtea, I. Casu, D. Comanescu. Steepest descent algorithm on orthogonal Stiefel manifolds. *arXiv:1709.06295*, 2017.
6. G. E. P. Box. Non-Normality and Tests on Variances. *Biometrika*, **40**, 318–335, 1953.
7. J. Blimberg. The Fermat-Weber location problem revisited. *Mathematical Programming*, **71**, 71–76, 1995.
8. E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theory* **56** (5), 2053–2080, 2009.
9. E. J. Candès, X. Li, Y. Ma, J. Wright. Robust Principal Component Analysis? *Journal of ACM* **58** (1), 1–37, 2009.
10. F. Castells, P. Laguna, L. Sornmo, A. Bollmann, J. M. Roig. Principal Component Analysis in ECG Signal Processing. *EURASIP Journal on Advances in Signal Processing*, 2007. <https://doi.org/10.1155/2007/74580>
11. F. Castells, C. Mora, J. J. Rieta, D. Moratal-Pérez, J. Millet. Estimation of atrial fibrillatory wave from single-lead atrial fibrillation electrocardiograms using principal component analysis concepts. *Medical and Biological Engineering and Computing*, **43** (5): 557–560, 2005.
12. J. D. Collins, W. T. Thomson. The eigenvalue problem for structural systems with statistical properties. *AIAA Journal*, **7** (4): 642–648, 1969.
13. P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, **47** (4): 547–553, 2009.
14. Y. Choi, J. Taylor, R. Tibshirani. Selecting the number of principal components: estimation of the true rank of a noisy matrix. *Annals of Statistics*, 2017.
15. C. Croux, G. Haesbroeck. Principal Component Analysis Based on Robust Estimators of the Covariance or Correlation Matrix: Influence Functions and Efficiencies. *Biometrika*, **87** (3): 603–618, 2000.
16. R. Durier. The Fermat-Weber Problem and Inner-Product Spaces. *Journal of Approximation Theory*, **58** (2): 161–173, 1994.
17. C. Eckart, G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, **1**, 211–218, 1936.
18. A. Edelman, T. A. Arias, S. T. Smith. The Geometry of Algorithms with Orthogonality Constraints. *SIAM Journal on Matrix Analysis and Applications*. **20** (2): 303–353, 1998.
19. C. Fraikin, K. Huper, P. Van Dooren. Optimization over the Stiefel manifold. *Proc. in Appl. Math. Mech.* **7** (1), 2007.
20. P. Geladi, B. Kowalski. Partial Least Squares Regression: A Tutorial. *Analytica Chimica Acta*. **185**: 1–17, 1986.
21. P. J. Huber, E. M. Ronchetti, Robust Statistics, 2nd Edition, March 2009.
22. Jolliffe I.T. Principal Component Analysis. *N.Y.: Springer-Verlag*, 2002.
23. R. Kramer. Chemometric Techniques for Quantitative Analysis. *New York: CRC Press.*, 1998.
24. R. A. Maronna, D. Martin, V. J. Yohai. Robust Statistics. *Chichester: Wiley*, 2006.
25. R. A. Maronna, V. J. Yohai. Robust Estimation of Multivariate Location and Scatter. *Encyclopedia of Statistical Sciences. New York: Wiley*, 589–596, 1998.
26. R. Muthukrishnan, E. D. Boobalan, R. Mathaiayn. MCD based Principal Component Analysis in Computer Vision. *International Journal of Computer Science and Information Technologies*, **5** (6), 8293–8296, 2014.

27. R. D. Neidinger. Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming. *SIAM Review*. **52** (3): 545–563, 2010. doi:10.1137/080743627.
28. Y. Nesterov. Introductory lectures on convex optimization: A basic course, volume 87 of Applied Optimization. *Kluwer Academic Publishers, Boston, MA*, 2004.
29. K. Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*. **2** (11): 559–572, 1901.
30. B. T. Polyak and M.V. Khlebnikov. Principal Component Analysis: Robust versions, *Automation and Remote Control* **78** (3): 490–506, 2017.
31. R. T. Sam. EM Algorithms for PCA and SPCA. *Advances in Neural Information Processing Systems (NIPS)*. 626–632, 1997.
32. Sleep In Mammals, Dataset, URL: <http://www.statsci.org/data/general/sleep.html>.
33. F. Torre, M. J. Black. Robust Principal Component Analysis for Computer Vision. *IEEE Information Theory Society Newsletter*, 2002.
34. E. Weiszfeld. Sur le point par lequel la somme des distances den points donnés est minimum. *Tohoku Mathematics Journal*, **43**, 355–386, 1937.
35. Z.Wen, W.Yin. A feasible method for optimization with orthogonality constraints. *Math. Progr.*, **142** (1): 397–434, 2013.
36. Wright J., Peng Y., Ma Y., Ganesh A., Rao S. Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices by Convex Optimization. *Proc. 23rd Ann. Conf. Neural Inform. Proc. Syst. (NIPS 2009). Vancouver, Canada*, 2080–2088, 2009.
37. Zhou Z., Li X., Wright J., Candès E.J., Ma Y. Stable Principal Component Pursuit. *Proc. IEEE Int. Symp. Inform. Theory (ISIT 2010). Austin, Texas*, 1518–1522, 2010.
38. H. Xu, C. Caramanis, S. Sanghavi. Robust PCA via Outlier Pursuit. *IEEE Trans. Inform. Theory*, **58** (5): 3047–3064, 2012.