Non-Negative Sparse PCA via Semi-Definite Programming

Aaron MAMANN[*] Guillaume LECUE [†]

CREST - August 2019

Abstract

Principal component analysis (PCA) is a well-known statistical method which enables to convert high-dimensional datasets into low-dimensional datasets. This dimension reduction allows to compare efficiently data observations between each other. Therefore, PCA turns out to be useful when it comes to Unsupervised Machine Learning (namely for clustering), outlier correction . . .

Adding sparsity makes the principal components easier to interpret as they would be linear combinations of fewer original variables.

Non-negative Sparse PCA has the particularity to project only on principal components whose all coordinates (with respect to the basis composed of the original variables) are non-negative — a quality in many fields including economics, bioinformatics (namely in the analysis of gene expression data to spike sorting in neural signal processing ([1])) and computer vision ([2]). Moreover, non-negativity participates in the sparsity.

1 Introduction

Our work consists in giving a solving method for Non-negative Sparse PCA via Semi-Definite Programming (SDP). To do so, we are going to use **two assumptions in order to simplify the solving method**. First, we will work under the "spiked covariance model". Second, each principal component $\beta = (\beta_j)_{j\in[[1,d]]} \in M_{d,1}(R^+)$ will follow the constraint $\sum_{j=1}^{d} \beta_j = 1$.

Non-negative Sparse PCA can be written as an optimisation problem:
$\beta^{(1)} = \arg\min_{\beta\in M_{d,1}(R^+)/\|\beta\|_2^2\leq 1, \|\beta\|_0 = k} -E\langle X,\beta\rangle^2$, where $\|\beta\|_0 = k$ ($k \in [[1,d]]$) means exactly k terms among the $(\beta_i)_{i\in[[1,d]]}$ are different from 0 and where $X = (X_i)_{i\in[[1,d]]}$ is a random vector with $E(X) = 0$. But we may notice, $-\langle E((1\ X)\otimes(1\ X)), (1\ \beta)\otimes(1\ \beta)\rangle = -1 - E\langle X,\beta\rangle^2$. Adding L^1-norm penalty, $\beta^{(1)} = \arg\min_{\beta\in M_{d,1}(R^+)/\|\beta\|_2^2\leq 1} -\langle E((1\ X)\otimes(1\ X)), (1\ \beta)\otimes(1\ \beta)\rangle + \lambda\sum_{i=1}^{d}|\beta_i|$, where $\lambda > 0$. But let us remind the former assumption we made : $\sum_{j=1}^{d}\beta_j = 1$.

[*]Research Intern at CREST, Student at ENSAE, contact : aaron.mamann@ensae.fr

[†]Researcher in Statistics and Machine Learning at CREST, Professor of Statistics and Machine Learning at ENSAE, contact :lecueguillaume@gmail.com, guillaume.lecue@ensae.fr

Therefore, our goal is to solve :

$$\underset{M=(1\ \beta)\otimes(1\ \beta),\sum_{j=1}^{d}\beta_j=1,\beta\in M_{d,1}(R^+)}{\arg\min} -\langle E((1\ X)\otimes(1\ X)),M\rangle \tag{1}$$

The two former assumptions will allow us to prove that the optimisation problem (1) is equivalent to the relaxed optimisation problem (2) (in the chapter below). This relaxation is useful as it leads to optimise in a well-known convex set (the set of positive semi-definite matrices) and therefore enables to use Semi-Definite Programs to resolve the Non-negative Sparse PCA. Moreover, this relaxation allows to optimise a linear function instead of a quadratic one.

2 Positive Semi-Definite Relaxation

In order to solve the optimisation problem (1) , we are going to introduce a relaxed version of this problem. To do so, we are going to prove in this chapter that the set (1) is equal to the set (2) below :

$$\underset{\substack{M\in M_{d+1}(R^+)/M\succeq 0\\ M_{1,1}=1,\sum_{k=2}^{d+1}M_{1,k}=1\\ M_{1,i}=\sum_{k=2}^{d+1}M_{k,i},\forall i\in[[2,d+1]]}}{\arg\min} -\langle E((1\ X)\otimes(1\ X)),M\rangle \tag{2}$$

Let $M\succeq 0,\exists N\in M_l(R),M=NN^\top$, so, with Cauchy-Schwarz's theorem we know : $\forall i,j\in[[1,l]],|\sum_{k=1}^{l}N_{i,k}N_{j,k}|\leq(\sum_{k=1}^{l}N_{i,k}^2)^{1/2}(\sum_{k=1}^{l}N_{j,k}^2)^{1/2}$, thus,

$$\forall M\succeq 0,\forall i,j\in[[1,l]],|M_{i,j}|\leq M_{i,i}^{1/2}M_{j,j}^{1/2} \tag{3}$$

Let us notice, for M $\in M_{d+1}(R^+)$:
$-\langle E((1\ X)\otimes(1\ X)),M\rangle$
$= -1+\sum_{(l,k)\in[[2,d+1]]^2,l\neq k}-E(X_{l-1}X_{k-1})M_{l,k}+\sum_{l\in[[2,d+1]]}-E(X_{l-1}^2)M_{l,l}$

We can also notice that : $(E(X_i,X_j))_{(i,j)\in[[1,d]]^2}=(cov(X_i,X_j))_{(i,j)\in[[1,d]]^2}$,
and as we assumed we are under the "spiked covariance model" we know :
$\exists v\in M_{d,1}(R^+),\theta>0,(cov(X_i,X_j))_{(i,j)\in[[1,d]]^2}=\theta vv^\top+Id=(E(X_i,X_j))_{(i,j)\in[[1,d]]^2}$
So, $\forall(l,k)\in[[2,d+1]]^2,l<k,-E(X_{l-1}X_{k-1})=-\theta v_{l-1}v_{k-1}\leq 0$.

Therefore, cf (3), if we set $A_1=\{M\in M_{d+1}(R^+)/M\succeq 0,\sum_{k=2}^{d+1}M_{1,k}=1,$
$M_{1,1}=1,M_{1,i}=\sum_{k=2}^{d+1}M_{k,i},\forall i\in[[2,d+1]]\}$, we can therefore deduce :
$\arg\min_{M\in A_1}-1+\sum_{(l,k)\in[[2,d+1]]^2,l\neq k}-E(X_{l-1}X_{k-1})M_{l,k}+\sum_{l\in[[2,d+1]]}-E(X_{l-1}^2)M_{l,l}$
$=\arg\min_{M\in A_1}\sum_{(l,k)\in[[2,d+1]]^2,l\neq k}-E(X_{l-1}X_{k-1})M_{l,l}^{1/2}M_{k,k}^{1/2}+\sum_{l\in[[2,d+1]]}-E(X_{l-1}^2)M_{l,l}$

Let us set, $A_2 = A_1 \cap \{M \in M_{d+1}(R^+)/M_{i,j} = M_{l,l}^{1/2} M_{k,k}^{1/2}, i,j \in [[2, d+1]]\}$.
$\arg\min_{M \in A_1} -\langle E((1\ X) \otimes (1\ X)), M\rangle = \arg\min_{M \in A_2} -\langle E((1\ X) \otimes (1\ X)), M\rangle$.

Let $M \in A_2$. As $M \succeq 0$, we have $\forall i \in [[1, d+1]], M_{i,i} \geq 0$, so without loss of generality we can write that :$\exists (\beta_i)_{i \in [[1,d]]} \in (R^+)^d, \forall i \in [[2, d+1]], M_{i,i}^{1/2} = \beta_{i-1}$.
As a result, $\exists \beta \in M_{d,1}(R^+), (M_{i,j})_{(i,j) \in [[2,d+1]]^2} = (\beta_i \beta_j)_{(i,j) \in [[1,d]]^2} = \beta\beta^\top$.
As, $\forall i \in [[2, d+1]], M_{1,i} = \sum_{k=2}^{d+1} M_{k,i}$, so, $\forall i \in [[2, d+1]], M_{1,i} = \beta_{i-1} \sum_{k=2}^{d+1} \beta_{k-1}$,
and, $\sum_{k=2}^{d+1} M_{1,k} = 1 = (\sum_{k=1}^{d} \beta_k)^2$, so, $(M_{1,i})_{i \in [[2,d+1]]} = (\beta_i)_{i \in [[1,d]]}$.

Thus,

$$\exists \beta \in M_{d,1}(R^+), M = \begin{pmatrix} 1 & \beta^\top & \\ \beta & \beta\beta^\top & \\ & & \end{pmatrix} = (1\ \beta) \otimes (1\ \beta)$$

As a result, we have proved the set (1) and (2) are equal.

3 Semi-Definite Programming

We have shown (in the chapter above) that solving the optimisation problem (1) and solving the relaxed version of it (2) is equivalent. Therefore, as we try to solve an optimisation problem on the set of positive semi-definite matrices, we have created Semi-Definite Programs (SDP). We have implemented two algorithms (in Python) in a notebook we have made. You can read this notebook here :

https://github.com/AaronMAMANN/NonNegative-Sparse-PCA

In the two algorithms (in the notebook), we have decided to reproduce the "spiked covariance model" in the data. Therefore, we have assumed that each observation $(X_i)_{i \in [[1,n]]}$ follows the same probability distribution than X where $X = (v^\top Y)v + Z$ where Z can be seen as a noise, $(v^\top Y)v$ being the random vector Y orthogonally projected on the vector space spanned by v, Z and Y being random vectors with d rows and 1 column, Z being independent of Y with $E(Z) = 0, E(ZZ^\top) = I_d$. So, we are under a "spiked covariance model" as $E(X) = \mu vv^\top + Id$ where $\mu = E[(vY)^2]$.

For the simulations of the algorithms, we had to choose a specific vector v, so we have chosen v (v=(1/k,...,1/k,0,...,0)) whose first k components are equal to 1/k. Let us observe we have selected a sparse vector v with non-negative components whose sum is equal to 1 exactly like in our optimisation problem (cf assumption in the Introduction). As a result, comparing v with the solution of our two algorithms is a good way to see if they work well.

As regards, the algorithm n^o1 we have used the python library CVXPY which consists in Convex Optimisation Programming and we have made use of the SDP option included this library. After having simulated the algorithm n^o1, let us observe that the computational complexity of this algorithm and its results are very satisfactory. Indeed, the distance (with the standard scalar product) between v and the first component is about 0.0697434 with a dataset of 500 variables and 10 000 observations and with v = (1/250,...,1/250,0,...,0) (v has 250 null coefficients). Let us notice the more we add observations the smaller the distance - between the first component and v - is.

When it comes to the algorithm n^o2, which also aims to solve our relaxed optimisation problem (2), we have made use of the Burer-Monteiro method which enables us to simulate Positive Semi-Definite Programs in polynomial time ([3]). In the optimisation problem (2), if we add the constraint $M = YY^\top$ where $Y \in M_{d+1,k}(R)$ (the set of k is going to be dealt with after), if Y is a rank-deficient local optimum, then $M = YY^\top$ is a global optimum ([3]). Therefore, in the algorithm n^o2, we are going to search for these Y rank-deficient local minimizers.
Moreover, if the search space of the optimisation problem (which we have called A_1 in the chapter two) is compact, we can set $k \sim \sqrt{2m}$, where k is the number of columns of Y and m is the number of constraints of the optimisation problem ([4],[5]). So we are going to show that the search space A_1 is compact. First, let us note that $S^+_{d+1}(R) = \bigcap\limits_{X \in R^{d+1}} \{M \in S_{d+1}(R), X^\top MX \geq 0\}$, as $S^+_{d+1}(R)$ is an intersection of preimages of a closed set under a continuous function, it is an intersection of closed sets, thus $S^+_{d+1}(R)$ is a closed set. The constraints we added in our optimisation problem are also preimages of closed sets by continuous functions. Therefore, the search space of our optimisation problem is closed. Furthermore, if $M \in A_1$, then $\sum_{k=2}^{d+1} M_{1,k} = 1$ and $M_{1,i} = \sum_{k=2}^{d+1} M_{k,i}$ (i $\in [[2, d+1]]$). As a result, $M_{i,j} \leq 1((i,j) \in [[1, d+1]]^2)$. So, $||M||_F \leq (d+1)^2$ ($|| \ ||_F$ is the Frobenius norm). So, the search space of our optimisation problem is a bounded set. Therefore, as the search space is a subset of $M_{d+1}(R)$ which is itself a finite-dimensional normed vector space and as the search space is closed and bounded, the search space is compact.
Thus, in the algorithm n^o2, we have set $k = \lfloor\sqrt{2m}\rfloor + 1$ (where k is the number of columns of Y). In this algorithm, we have used the optimisation program from the library Scipy named Optimize.minimize which enables to deal with non-linear programming. However, the results of the function depend strongly on the first guess we give. Therefore, in order to give a good first guess, we have implemented an optimisation program based on a conjugate gradient method to build a trust-region. Nevertheless, there are still little problems in the algorithm to be operational namely some operations which strongly increase its computational complexity.

References

[1] Andrea Montanari and Emile Richard, *Non-negative Principal Component Analysis: Message Passing Algorithms and Sharp Asymptotics, June 2014*

[2] Ron Zass and Amnon Shashua, *Nonnegative Sparse PCA, 2006*

[3] Samuel Burer and Renato D.C. Monteiro, *A Nonlinear Programming Algorithm for Solving Semidefinite Programs via Low-rank Factorization, March 2001*

[4] G. Pataki, *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues, Mathematics of operations research, 23(2):339–358, 1998*

[5] A.I. Barvinok, *Problems of distance geometry and convex properties of quadratic maps. Discrete Computational Geometry, 13(1):189–202, 1995.*