Non Negative Matrix Factorization

Marc Chachuat - Pierre Colombo - Vincent Petri

May 16, 2017

In this paper we study an application of convex optimization to the problem of nonnegative matrix factorization. We analyze and experiment a new method presented by V. Krishnamurthy and A. d'Aspremont in [1]. Classical non-convex algorithms used to solve nonnegative matrix factorization seek for a local optimum. They are therefore sensitive to the choice of the starting point, and the computation can be arbitrary long. To address these issues, V. Krishnamurthy and A.d'Aspremont introduce a convex approximation of this problem, and solve it using convex optimization. We will detail this last strategy and compare its performance to two non-convex algorithms : alternating least square method and multiplicative update.

Our paper is organized as follow : First we present the matrix nonnegative factorization problem. Then we give theoretical explanations on the new method. In the third part, we present our pseudo code. The last part is dedicated to the comparison of performances.

1 PRESENTATION OF THE FACTORIZATION PROBLEM

1.1 PROBLEM DEFINITION

Given a matrix $A \in R^{m \times n}$, the non negative matrix factorization problem can be written as follow:

$$\begin{aligned} \underset{U,V}{\text{minimize}} \quad & l(A, UV^T) \\ \text{subject to} \quad & U, V \geqslant 0 \end{aligned}$$

with $U \in R^{m*k}$ and $V \in R^{n*k}$. For a matrix X, $X \geq 0$ means here that all coefficients of X are non negative. Therefore, U and V have only non negative coefficients.
l is a loss function and k represents a maximal rank that we choose as parameter with $k \leqslant min(m,n)$.
Note that if $k \geq n$ and $A \geq 0$, this problem becomes trivial as AI is a solution.
This problem looks rather simple but it is a NP-hard problem. Moreover, the non-negativity constraint cannot be lifted as it is required by the data themselves, in most application where this type of factorization is used.

1.2 EXAMPLE OF APPLICATION

Non negative matrix factorization is a widely used technique for learning parts-based representations of a dataset. For this reason it has a lot of applications in all domains that can use machine learning. For instance, it is used in text mining as a way to find the topic of a text or to track trends in a corpus or in social media. Image analysis, bioinformatics and computer vision use this technique as well.

1.3 "CLASSICAL" ALGORITHMS

According to our knowledge, three main classes of algorithms are used to solve this problem.

1.3.1 Multiplicative update

In this first method, we update iteratively each component of U and V, using rules which depend on the loss function. For instance, with the Mean Squared Error, the update rules are defined as follow:

$$V_{ij}^{+} = V_{ij}\frac{(U^T A)_{ji}}{(U^T U V^T)_{ji}} \text{ and } U_{ij}^{+} = U_{ij}\frac{(AV)_{ij}}{(UV^T V)_{ij}}$$

1.3.2 Alternating least squares

This second method is used with the Mean Square Error as loss function. When U (respectively V) is fixed, the minimization problem is reduced to a linear least square problem in V (respectively U). This means that we can try to solve the problem by alternating between the least-squares problem in U and V. (Progress is "guaranteed" as we decrease the objective at each iteration)

1.3.3 Gradient descent

The gradient descent is also widely used to solve this problem, even if it is non-convex. (In convex problems, gradient descent is very popular because local minima are also global). As the problem is non-convex, there is a risk that the minimum attained by the algorithm is only local. However, to reduce this risk, it is possible to use a stochastic gradient descent.

2 Theoretical Explanations

2.1 Convex Approximation

In this section we present the convex approximation of the nonnegative matrix factorization, as stated in [1].

2.1.1 Hypothesis

Because of time constraints, we decided to restrict our study to the symmetric factorization.
In this case, we will assume that $A \in S_n(\mathbb{R})$, and our problem will be simplified to :

$$\begin{aligned} &\underset{U}{\text{minimize}} && l(A, UU^T) \\ &\text{subject to} && U \geqslant 0 \\ &\text{with} A \in S_n(\mathbb{R}) \end{aligned}$$

Let us give an intuition behind this simplification (this isn't a formal proof) :
This simplification can be seen as a consequence of the spectral theorem. All real symmetric matrices can be expressed as UDU^T where D is diagonal and U contains the eigenvectors of the initial matrice. When the eigenvalues are positive, the matrix can even be factorized into UU^T (but here, the matrices U are allowed to have negative coefficients).
Therefore, it is natural to look for a non negative factorization of the same shape, i.e. when U and V are identical.

Definition 1: We define $\geqslant$ in the following way : $\forall U \in R^{m\times n}, U \geqslant 0$ if and only if all components of U are positives.

Loss function: According to what is recommended in [1], we use the Kullback-Leibler Divergence (KL) as loss function.

$$\text{KLloss}(X, Y) = \sum_{i,j=1}^{N} (X_{i,j} * log(X_{i,j}/Y_{i,j}) + Y_{i,j} - X_{i,j})$$

2.1.2 Definitions & Results

Let us start with some definitions and one fundamental result. This result enables us to show the correctness of the convex restriction.

Definition 2: Completly positive matrices We call completly positive matrices, all matrices $A \in S_n(\mathbb{R})$ such that A is an exact solution of the previous problem. A can be written as $A = UU^T$ with $U \geqslant 0$

Definition 3 : Hadamard exponential For every matrix X, the Hadamard exponential of X, denoted $(exp_H(X))$ is the componentwise exponential of X.

Theorem 1: If $X \in S_n(\mathbb{R})^+$ (X is a symmetric positive semidefinite matrix), then $exp_H(X)$, the Hadamard exponential of

X, is completely positive.

For the proof of this theorem, the reader can refer to [1].

2.1.3 Convex Restriction

The previous theorem allow us to safely restrict our previous approximation by introducing the change of variable : $A_{i,j} = exp(X_{i,j})$. (This is a restriction because even if $exp_H(X)$ is completely positive for $X \in S_n(\mathbb{R})^+$, we don't know if all completely positive matrices can be expressed as an Hadamard exponential).

We obtain the convex optimization problem in $X \in S_n(\mathbb{R})^+$:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \sum_{i,j=1}^{N} (A_{i,j} * (log(A_{i,j}) - X_{i,j}) + exp(X_{i,j}) - A_{i,j}) \\ \text{subject to} \quad & X \in S_n(\mathbb{R})^+ \end{aligned}$$

Claim : This formulation is a convex restriction of the problem

1. It is a restriction : Suppose $X \in S_n(\mathbb{R})^+$ is a solution to the previous problem. According to theorem 1, $exp_H(X)$ is completely positive.Hence if we can factor $exp_H(X)$ in $U.U^T$, any feasible solution of our problem will give us a feasible solution to the initial problem. The next section will be dedicated to this factorization.
2. The objective above is convex as a sum of convex and linear functions ($X \to exp(X_{i,j})$ is convex as the combination of a projection – linear function – and an exponential).
   Moreover, the cone of positive semi definite matrices $S_n(\mathbb{R})^+$ is a convex set. We minimize a convex function over a convex set, this is a convex optimization problem.

2.2 Factorization of the exponential

In this section, we present the method used by the authors of [1] to decompose approximatively the Hadamard exponential of a matrix X into UU^T.
Then, we explain why we will need to restrict our study to matrices A with small eigenvalues.

2.2.1 Hadamard exponential decomposition

In this section, we will denote by AoB the piecewise product of the two matrices A and B (or of the two vectors A and B).
Let $X \in S_n(\mathbb{R})^+$. We want to find $U \in \mathbb{R}^{nxk}$ for some $k \in \mathbb{N}^*$ such that $exp_H(X) = UU^T$. The following Lemmas will help us to find an approximate solution :

Lemma 1 : If $A = \sum_{i=1}^{k} a_i a_i^T$ and $B = \sum_{j=1}^{l} b_j b_j^T$
Then, $AoB = \sum_{i=1}^{k} \sum_{j=1}^{l} (a_i o b_j)(a_i o b_j)^T$.

Lemma 2 : As a consequence of Lemma 1, if A and B are two completely positive matrices, whose factorizations are known, AoB is a completely positive matrix whose matrix factorization depends explicitely on the factorizations of A and B.

In this case, note that if A (respectively B) is factorized in UU^T (resp VV^T) where U (resp V) has k (resp. l) columns, then it exists an explicit decomposition of AoB in WW^T where W has $k \times l$ columns.

Using the two previous lemmas, we can immediatly deduce a strategy to compute the matrix factorization of $exp_H(X)$:

1. As X is p.s.d, we can decompose it into $X = \sum_{k=1}^{n} \lambda_k x_k x_k^T$
2. Then $exp_H(X) = \prod_{k=1}^{n} exp(\lambda_k x_k x_k^T)$ where the product is a piecewise product between matrices.
3. To compute the factorization of $exp_H(X)$ it is then sufficient to compute the matrix factorization (into completely positive matrices) of each of the $exp(\lambda_k x_k x_k^T)$ and to deduce iteratively the global factorization of $exp_H(X)$ from the rule of lemma 1.

Factorization of $exp(\lambda_k x_k x_k^T)$
Let $v \in \mathbb{R}^n$, we want to factorize $exp(vv^T)$ into UU^T with U completely positive. Note that this solve the problem presented above when we fix $v = \sqrt{(\lambda_k)} x_k$.
We re-explain here the method presented in [1]:

Let $M = \max_{i=1..n} |v_i|$ then, $exp_H(vv^T)_{i,j} = exp(v_i v_j) = exp(-M^2 + (M + x_i)(M + x_j) - M(x_i + x_j))$
Therefore, with $y = M1 + v$ and $z = exp(-Mv)$, we have : $exp_H(vv^T) = exp(-M^2) exp_H(yy^T) o zz^T$.

Then as $exp(-M^2) > 0$, and zz^T is a completely positive matrix, from Lemma 1 and Lemma 2 we just have to find the matrix factorization of $exp_H(yy^T)$.
Intuitively, y has only positive coefficients and $exp_H(yy^T)$ is an infinite sum of terms of the form $\frac{(yy^T)^{(o)k}}{!k}$. Where $(yy^T)^{(o)k}_{i,j} = (y_i y_j)^k$. Therefore, it is a matrix factorization of two infinite matrices with only positive coefficients. We will approximate it by the k first terms of the sum.
To sum up,as suggested in [1] we approximate $exp_H(yy^T)$ by UU^T where $U \in \mathbb{R}^{nxk}$ and the k-th column of U is $U_k = \frac{y^k}{\sqrt{!k}}$.
This whole process is detailed in the pseudo code of the next part.

2.2.2 An approximation issue

Note that the number of columns in the final matrix U obtained iteratively by appliying lemma 1 to the $exp(\lambda_k x_k x_k^T)$, is equal to k^n (or k^m if we keep only the m more significant eigenvalues in the decomposition of X). But it would be useless to factorize $exp_H(X)$ in UU^T with U having a huge number of columns. That's why we need to restrict ourselves to small values of k.
Then, the problem is that $exp_(yy^T)$ won't be correctly approximated as soon as y has big values (Indeed, the exponential of a real r is correctly approximated by its very first terms if and only if r is close to 0). But the values of y are proportional to $\sqrt{\lambda}$ for λ eigenvalue of X
We will then have to reduce our method to $X \in S_n(\mathbb{R})^+$ with very small eigenvalues ($\lambda \leq 1$).

As this restricition wasn't mentionned in [1], we justified it by empirical results, that will be presented in the last part of this paper.

3 Algorithm

In the previous section, we have :

1. introduced a convex approximation of the nonnegative matrix factorization problem, when the matrix to factorize is symmetric.

2. presented an explicit method to recover the factorization, starting from the solution of the convex approximation.

This third part is dedicated to the algorithms we used to test these theoretical developments. In the first section, we present our global pseudo-code. In section 2, we detail the projection on the cone of semi definite matrices. Finally, in section 3, we explicit the gradient of the KL- divergence.
Note that the codes corresponding to these pseudo-codes will be provided with this paper.

3.1 Global Pseudo Code

Input : A a symetric semi-definite matrix of size $N * N$ with small eigenvalues (We assume that with this constraint, the eigenvalues of the optimal X will be very small. As a consequence, the matrix factorization of $exp_H(X)$ should be accurate)
Output : U a symetric semi-definite matrix of size $N * N$ such that $U.U^T \approx A$

3.1.1 Solve the convex optimization problem

Algorithm 1 Use a gradient descent algorithm to solve the program (2) and find X :

procedure SOLVE (X)
start from $X = 0 \in R^{N*N}$
$X^{k+1} = X^k - \text{stepsize} * \|\nabla_X(loss(A, X^k))\|$
k = k+1
repeat until $\|\nabla_X(loss(A, X^{k+1}))\| <$ precision
return X
end procedure

Algorithm 2 Factorize $exp_H(X)$

procedure FACTORIZE($exp_H(X)$)
Compute the eigenvalue decomposition : $X = \sum_{i=1}^{n} \lambda_i x_i x_i^T$
Decompose each factor, $exp_H(v_i v_i^T) = \exp(-M^2) exp_H(y_i y_i^T) o z_i z_i^T$ where $v_i = \sqrt{(}\lambda_i) x_i$
approximate $exp_H(y_i y_i^T)$ as $\sum_{j=0}^{k} \frac{(y_i y_i^T)^j}{!j}$
Collect all the terms above using the chain rule (see previous part) to compute U such that $exp_H(X) \approx UU^T$
return (U)
end procedure

3.1.2 FACORIZE THE HADAMARD EXPONENTIAL

3.2 PROJECTION OF X ON THE CONE OF SEMI DEFINITE MATRIX

The projection of X on the cone of semi definite matrix is the solution to the following convex program:

$$\begin{aligned} \underset{B}{\text{minimize}} \quad & \|A - B\|_F \\ \text{subject to} \quad & B \in S_n(\mathbb{R})^+ \end{aligned}$$

3.3 GRADIENT OF THE KULLBACK-LEIBLER DIVERGENCE

The gradient of the KL loss can be easily computed :

$$\nabla_X(loss(A,X) = \begin{pmatrix} exp(X_{11}) - A_{11} & exp(X_{12}) - A_{12} & ... & exp(X_{1N}) - A_{1N} \\ ... & ... & ... & ... \\ exp(X_{N1}) - A_{N1} & exp(X_{N2}) - A_{N2} & ... & exp(X_{NN}) - A_{NN} \end{pmatrix}.$$

4 PERFORMANCE EVALUATION

This last part is dedicated to the results we found by running the algorithms introduced above. We first present the empirical results that justify the restriction to matrices with small eigenvalues (as mentionned in Part 2, section 2). Then, we compare the performance of our algorithm to those of the classical algorithms presented in part 1.

4.1 RESTRICTION TO SMALL-EIGENVALUE MATRICES

As mentionned in the second part of this paper, our theoretical analysis suggests that the approximate factorization of the Hadamard exponential will perform poorly when the eigenvalues of the argument matrix are not small enough.
We show below empirical results that support our theoretical analysis.

4.1.1 DESCRIPTION OF OUR EXPERIMENT

For differents values of λ_{max},

- we generate $n = 100$ samples of matrices $X \in \mathbb{R}$, whose maximal eigenvalue is lower than λ_{max}. For each of them :
  - We compute the the approximate matrix factorization UU^T of $exp_H(X)$ (setting our choice of $k = 5$ for the approximation of the exponential as a sum – see part 2 for more details) .
  - We compute the error : $\|UU^T - exp_H(X)\|_{Frob}$
- Then we compute the average error over the n samples.

With this process, we asociate to each λ_{max} an average error when computing the matrix factorization of $exp_H(X)$, where X has eigenvalues upper bounded by λ_{max}.
According to our theoretical analysis, we should observe the error exploding as λ_{max} increases.

4.1.2 RESULTS OF OUR EXPERIMENT

The plot above illustrates the result of our experiment. One can see that the observed behaviour sticks to the expected one. As a consequence, it is correct and necessary to restrict our study to cases where X is susceptible to have low eigenvalues, i.e to matrices A with low eigenvalues.

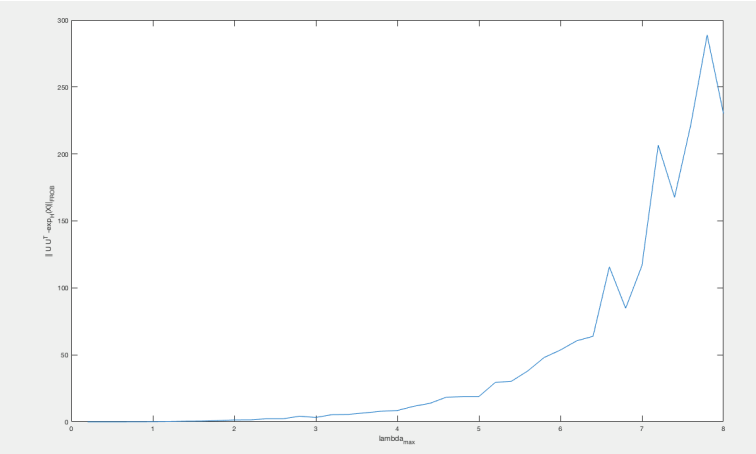

Figure 4.1: Factorization error as a function of the maximal eigenvalue of X

4.2 Performance evaluation

4.2.1 Protocol

We compare the performance of three algorithms :

1. the one describe above
2. the alternative least square algorithm
3. the multiplicative update algorithm

by measuring for each of them, the Frobenius norm of the error commited ($\|A - UU^T\|_{frob}$ where A is the input and U the output of the algorithms).

More precisely, we tried to compute this error as a function of the distance to an Hadamard exponential (that is, as a function of the degree of approximation of the convex formulation).
We follow a protocol similar to the one of the previous section.
Let fix $B = \begin{pmatrix} -2 & -7 \\ -7 & 0 \end{pmatrix}$
For a given range of values of epsilon,

1. For a given number of samples :
   - We generate an Hadamard Exponential A.
   - We perturb it by adding a term $\epsilon * B$
   - For each of the three methods, we compute the factorization of the resulting matrix.
   - For each of the three methods, we compute the error committed.
2. For each of the three methods, we compute the average (over the samples) error.

Then, we have associated to each epsilon, three average errors (one for each method) committed by the algorithms, when the input is at distance at most $\epsilon * \|B\|_{Frob}$ of an Hadamard exponential.

Applying this protocol, we expect to see the error of the convex method increase with epsilon (because the convex approximation of the problem becomes less precise when epsilon increases).

4.2.2 Results

The figure belows presents the results we obtained by following the previous protocol.

From these plots, we can make some observations.

First, the behaviour of the convex method sticks to the one expected. The error increases as epsilon increases.

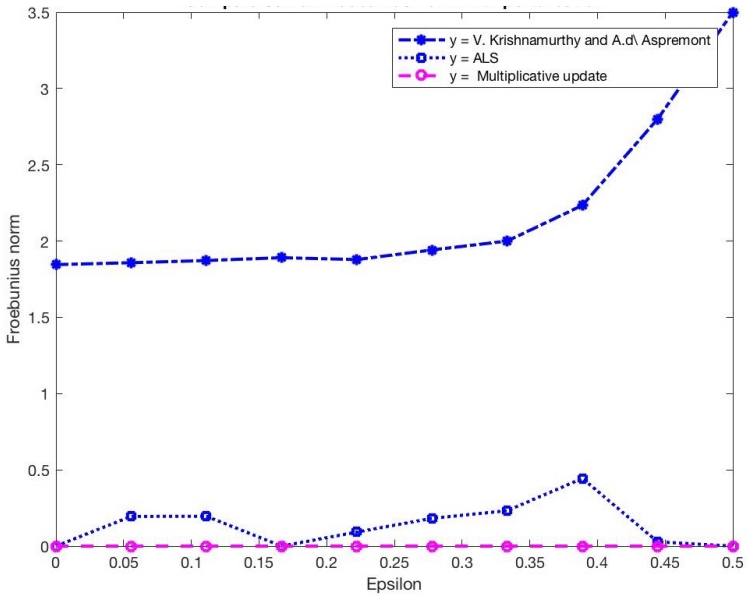

Figure 4.2: Factorization error as a function of the maximal distance to an Hadamard exponential

Moreover, for all epsilon, the error committed by the convex method is significantly higher than the error committed by the other methods.
We can indicate a few possible explanations to this low relative performance:

- Despite the restriction to matrices with small eigenvalues, it is possible that the approximation of the exponential (see theoretical part for more details) still induces a non negligible error.
- The first instance restriction to a factorization of the shape UU^T instead of UV^T may be a very strong restriction.

A last remark is that in addition to committing a higher error, the convex method is very slow (with our implementation) in comparison to the two other methods. Therefore, it doesn't seem, at first sights, of a practical interest.

5 Conclusion

To conclude, we have studied a very interesting method, which combines algebra and convex optimization to tackle a non convex problem.
Thanks to this project, we have become aware of the strong capacity of convex optimization to approximate problems of all kinds.
In a theoretical point of view, we have been able to re-use the notions seen in the CS454 lectures, and they were sufficient to gave us a good understanding of the paper. Moreover, in the theoretical analysis, we identified an important restriction on the input, restriction that wasn't mentioned in the paper of [1].
In a practical point of view, this work enabled us to use CVX tools. Elaborating and realizing two experiments, we stated two conclusions. First, we confirmed that the restriction mentioned above should be worked deeper. Then, we observed poor relative performance of the new method in terms of error committed, as well as running time. Therefore, if this method is innovative and interesting in a theoretical point of view, it doesn't seem of a practical interest to us (according to our modest experiments).

References

[1] Alexandre d'Aspremont & Vijay Krishnamurthy Convex Algorithms for Nonnegative Matrix Factorization