An Alternating Direction Algorithm for Nonnegative Matrix Factorization

Yin Zhang (张寅)

Department of Computational
and Applied Mathematics
Rice University, Houston, TX 77005

January 19, 2010

Abstract

We extend the classic alternating direction method for convex optimization to solving the non-convex, nonnegative matrix factorization problem and conduct several carefully designed numerical experiments to compare the proposed algorithms with the most widely used two algorithms for solving this problem. In addition, the proposed algorithm is also briefly compared with two other more recent algorithms. Numerical evidence shows that the alternating direction algorithm tends to deliver higher-quality solutions with faster computing times on the tested problems. A convergence result is given showing that the algorithm converges to a Karush-Kuhn-Tucker point whenever it converges.

1 Introduction

Nonnegative matrix factorization (NMF) is a dimension reduction and clustering technique for data analysis, most suitable for data that are innately additive mixtures of nonnegative components such as concentrations or intensities. It appears that NMF was first used by Paatero and his coworkers [15, 16, 17] in areas of environmental science. The 1999 paper of Lee and Seung [12] in the journal *Nature* has greatly popularized the use and research of this technique. To this date, there is already a vast literature[1] on the applications of NMF in various areas and on numerical algorithms for solving various NMF-related models (see, for example, the survey paper [1], two recent books [3, 4] and the references thereof).

The purpose of the present paper is to introduce yet another algorithm based on an extension of the classic alternating direction approach for solving a fundamental NMF model, and to present numerical evidence showing that on a range of tested problems the proposed algorithm brings a clearly improved performance in comparison to some existing state-of-the-art algorithms. A simple result on convergence is obtained showing that the algorithm can only converge to Karush-Kuhn-Tucker (KKT) points.

1.1 The NMF Problem

Given a matrix $M \in \mathbb{R}^{m\times n}$ of nonnegative entries, the nonnegative matrix factorization (NMF) problem is to approximate M by (or possibly decompose M into) a product of two lower-rank nonnegative matrices $X \in \mathbb{R}^{m\times k}$ and $Y \in \mathbb{R}^{k\times n}$ where $k < \min(m, n)$ (usually $k \ll \min(m, n)$ in applications). A more precise name for this problem is approximative nonnegative matrix factorization (ANMF), but the acronym NMF seems to have been accepted as the *de facto* standard name.

[1]An online search in Google Scholar using the key phrase "nonnegative matrix factorization" (within quotes) returned about 1580 entries on January 19th, 2010.

A fundamental model in NMF utilizes the least squares cost function to measure the closeness of matrices, resulting in the following standard NMF problem

$$\min_{X,Y} f(X,Y) \triangleq \frac{1}{2}\|XY - M\|_F^2 \text{ s.t. } X \geq 0,\ Y \geq 0, \tag{1}$$

where $\|\cdot\|_F$ is Frobenius norm, and the inequalities are component-wise. Other cost functions, particularly the Kullback-Leibler divergence, have also been used in NMF. Moreover, additional regularization terms in cost functions have been used in order to encourage desirable properties such as sparsity. In this paper, we will only focus on solving problem (1). Specifically, we propose the use of an alternating direction method (ADM) approach for solving problem (1). It should be clear that similar algorithms can also be derived for other models or cost functions from this ADM approach.

It is well understood that problem (1) is non-convex that generally permits non-global local minima. It is also clear that a given objective value $f(X,Y)$ can be attained by infinitely many different pairs of (X,Y) due to the invariance of the product XY under the transformation $(X,Y) \to (XD, D^{-1}Y)$ as long as nonnegativity is maintained (for example, D can be any positive diagonal matrix). The stationarity conditions for (1) can be written as

$$F(X,Y) \triangleq \begin{bmatrix} \min\left(X^T, Y(XY-M)^T\right) \\ \min\left(Y, X^T(XY-M)\right) \end{bmatrix} = 0, \tag{2}$$

where the minimums are taken component-wise, which is a condensed form for the more explicit conditions

$$\begin{array}{ll} X \odot (XY-M)Y^T = 0, & X, (XY-M)Y^T \geq 0 \\ Y \odot X^T(XY-M) = 0, & Y, X^T(XY-M) \geq 0, \end{array} . \tag{3}$$

where $\odot$ denotes component multiplications. A pair (X,Y) is called a Karush-Kuhn-Tucker (KKT) point of problem (1) if it satisfies (2) or (3). In our case, a KKT point can be either a local (or global) minimum or a saddle point, but not a local maximum unless both $(XY-M)Y^T = 0$ and $X^T(XY-M) = 0$ which is highly unlikely.

1.2 Most widely used algorithms for NMF

The first method used for solving problem (1) seems to be the alternating least squares (ALS) algorithm utilized by Paatero and Tapper in 1994 [15]. It minimizes the least squares cost function with respect to either X or Y, one at a time, while fixing the other and disregarding nonnegativity, and then sets any negative entries to zero after each least squares step. The scheme can be written as the following updates:

$$X_+ = \mathcal{P}_+\left(MY^T(YY^T)^\dagger\right), \tag{4a}$$

$$Y_+ = \mathcal{P}_+\left((X^TX)^\dagger X^TM\right), \tag{4b}$$

where the superscript "†" denotes pseudo-inverse, and $\mathcal{P}_+$ denotes the projection onto the set of nonnegative matrices of appropriate sizes; i.e., for a matrix X

$$\mathcal{P}_+(X) \triangleq \max(0, X) \tag{5}$$

where the maximum is taken component-wise. This algorithm is still widely used today.

Another vastly popular method for NMF is the multiplicative updating (or simply Mult) method proposed by Lee and Seung [13]. It can be written as

$$X_+ = X \odot [(MY^T) \oslash (XYY^T + \epsilon)], \tag{6a}$$

$$Y_+ = Y \odot [(X^TM) \oslash (X^TXY + \epsilon)], \tag{6b}$$

where $\oslash$ denotes component divisions and a small number $\epsilon > 0$ is added to guard against division by zero. When started from nonnegative initial guesses, the iterates will remain nonnegative throughout iterations. By all indications, this algorithm appears to have been the most widely used solution method in NMF by far.

The popularity of ALS and Mult algorithms among practitioners is perhaps attributable in part to their remarkable simplicity and to the fact that they do not require setting of algorithmic parameters, beside the fact that they do produce reasonable results in practical applications within a reasonable amount of time. Both of the above two algorithms have been implemented in the Matlab Statistics Toolbox. In fact, they are the only two algorithms implemented in the Statisitcs Toolbox (up to version V7.2 (R2009b)). The Matlab documentation states that "*In general, the 'als' algorithm converges faster and more consistently. The 'mult' algorithm is more sensitive to initial values, which makes it a good choice when using 'replicates' to find W and H from multiple random starting values.*" Here W and H are, respectively, X and Y in our notation.

The theoertical properties of the ALS and Mult algorithms, without modifications, are still not well understood. For instance, it is not known whether or not the algorithms converge to a stationary point when convergence does occur. Numerical stability is a known issue for ALS since X or Y can, and often does, become numerically rank deficient during iterations.

1.3 Organization

This paper is organized as follows. In Section 2, we propose a new algorithm for NMF by extending the classic alternating direction method for convex optimization. A simple convergence result for the proposed algorithm is given in Section 3. Section 4 contains several sets of computational results cpmparing the proposed algorithm with several state-of-the-art algorithms. Finally, we provide some concluding remarks in Section 5.

2 Alternating Direction Algorithm for NMF

2.1 The classic ADM approach

In a finite-dimensional setting, the classic alternating direction method (ADM) is for solving structured convex programs of the form

$$\min_{x\in\mathcal{X},y\in\mathcal{Y}} f(x)+g(y) \;\text{ s.t. } Ax+By=c, \tag{7}$$

where f and g are convex functions defined on closed convex subsets $\mathcal{X}$ and $\mathcal{Y}$ of finite-dimensional spaces, respectively, A, B and c are matrices and vector of appropriate sizes. The augmented Lagrangian function of (7) is

$$\mathcal{L}_A(x,y,\lambda)=f(x)+g(y)+\lambda^T(Ax+By-c)+\frac{\beta}{2}\|Ax+By-c\|_2^2, \tag{8}$$

where λ is a Lagrangian multiplier vector and $\beta>0$ is a penalty parameter.

The classic alternating direction method [7, 6] is an extension of the augmented Lagrangian multiplier method [9, 18, 19], It performs one sweep of alternating minimization with respect to x and y individually, then updates the multiplier λ; that is, at the iteration k an ADM scheme executes the following three steps: given (x^k, y^k, λ^k),

$$x^{k+1} \;\leftarrow\; \operatorname*{argmin}_{x\in\mathcal{X}} \mathcal{L}_A(x,y^k,\lambda^k), \tag{9a}$$

$$y^{k+1} \;\leftarrow\; \operatorname*{argmin}_{y\in\mathcal{Y}} \mathcal{L}_A(x^{k+1},y,\lambda^k), \tag{9b}$$

$$\lambda^{k+1} \;\leftarrow\; \lambda^k+\gamma\beta(Ax^{k+1}+By^{k+1}-c), \tag{9c}$$

where $\gamma\in(0,1.618)$ is a step length. It is worth noting that (9a) only involves $f(x)$ in the objective and (9b) only $g(y)$, whereas the classic augmented Lagrangian multiplier method [9, 18, 19] requires a joint minimization with respect to both x and y; i.e., replacing steps (9a) and (9b) by

$$(x^{k+1},y^{k+1}) \;\leftarrow\; \operatorname*{argmin}_{x\in\mathcal{X},y\in\mathcal{Y}} \mathcal{L}_A(x,y,\lambda^k),$$

which involves both $f(x)$ and $g(y)$ and could become much more expensive. A convergence proof for the above ADM algorithm can be found in [5].

2.2 ADM Extension to NMF

To facilitate an efficient use of alternating minimization, we first introduce two auxiliary variables U and V, and consider the following equivalent model,

$$\min_{X,Y,U,V} \frac{1}{2}\|XY - M\|_F^2 \text{ s.t. } X - U = 0,\ Y - V = 0,\ U \geq 0,\ V \geq 0, \tag{10}$$

where $U \in \mathbb{R}^{m\times k}$ and $U \in \mathbb{R}^{k\times n}$. The augmented Lagrangian function of (10) is

$$\begin{gathered} \mathcal{L}_A(X,Y,U,V,\Lambda,\Pi) = \tfrac{1}{2}\|XY - M\|_F^2 + \\ \Lambda \bullet (X - U) + \Pi \bullet (Y - V) + \tfrac{\alpha}{2}\|X - U\|_F^2 + \tfrac{\beta}{2}\|Y - V\|_F^2, \end{gathered} \tag{11}$$

where $\Lambda \in \mathbb{R}^{m\times k}$, $\Pi \in \mathbb{R}^{k\times n}$ are Lagrangian multipliers and $\alpha, \beta > 0$ are penalty parameters for the constraints $X - U = 0$ and $Y - V = 0$, respectively, and for matrices A and B of the same size the scalar product "$\bullet$" is the sum of all element-wise products, i.e., $A \bullet B = \sum_{i,j} a_{ij}b_{ij}$.

The alternating direction method (ADM) for (10) is derived by successively minimizing the augmented Lagrangian function $\mathcal{L}_A$ with respect to X, Y, U and V, one at a time while fixing others at their most recent values, and then updating the multipliers after each sweep of such alternating minimization. The introduction of the two auxiliary variables U and V makes it easy to carry out each of the alternating minimization steps. Specifically, these steps can be written in a closed form,

$$X_+ = \left(MY^T + \alpha U - \Lambda\right)(YY^T + \alpha I)^{-1}, \tag{12a}$$
$$Y_+ = (X_+^T X_+ + \beta I)^{-1}\left(X_+^T M + \beta V - \Pi\right), \tag{12b}$$
$$U_+ = \mathcal{P}_+(X_+ + \Lambda/\alpha), \tag{12c}$$
$$V_+ = \mathcal{P}_+(Y_+ + \Pi/\beta), \tag{12d}$$
$$\Lambda_+ = \Lambda + \gamma\alpha(X_+ - U_+), \tag{12e}$$
$$\Pi_+ = \Pi + \gamma\beta(Y_+ - V_+), \tag{12f}$$

where the subscript "+" is used to denote iterative values at the new iteration. Since the involved inverse matrices are both $k \times k$, the corresponding linear systems are relatively inexpensive for $k \ll \min(m,n)$. In this case, the dominant computational tasks at each iteration are the matrix multiplications MY^T and X^TM, together requiring about $4rmn$ arithmetic operations (scalar additions and multiplications).

3 Convergence to KKT Points

In this section, we provide a partial result on the convergence of the proposed ADM algorithm. To simplify notation, let us define the sextuple

$$Z \triangleq (X, Y, U, V, \Lambda, \Pi).$$

A point Z is a KKT point of problem (10) if it satisfies the KKT conditions for problem (10):

$$(XY - M)Y^T + \Lambda = 0, \tag{13a}$$
$$X^T(XY - M) + \Pi = 0, \tag{13b}$$
$$X - U = 0, \tag{13c}$$
$$Y - V = 0, \tag{13d}$$
$$\Lambda \leq 0 \leq U,\ \Lambda \odot U = 0, \tag{13e}$$
$$\Pi \leq 0 \leq V,\ \Pi \odot V = 0. \tag{13f}$$

Proposition 1. *Let $\{Z_k\}_{k=1}^{\infty}$ be a sequence generated by the ADM algorithm (12) that satisfies the condition*

$$\lim_{k\to\infty}(Z_{k+1}-Z_k)=0. \tag{14}$$

Then any accumulation point of $\{Z_k\}_{k=1}^{\infty}$ is a KKT point of problem (10). Consequently, any accumulation point of $\{(X_k,Y_k)\}_{k=1}^{\infty}$ is a KKT point of problem (1).

Proof. We can rearrange the ADM update formulas in (12) into

$$\begin{aligned}
(X_+-X)(YY^T+\alpha I) &= -\left((XY-M)Y^T+\alpha(X-U)+\Lambda\right), && \text{(15a)}\\
(X_+^TX_++\beta I)(Y_+-Y) &= -\left(X_+^T(X_+Y-M)+\beta(Y-V)+\Pi\right), && \text{(15b)}\\
(U_+-U) &= \mathcal{P}_+(X_++\Lambda/\alpha)-U, && \text{(15c)}\\
(V_+-V) &= \mathcal{P}_+(Y_++\Pi/\beta)-V, && \text{(15d)}\\
\Lambda_+-\Lambda &= \gamma\alpha(X_+-U_+), && \text{(15e)}\\
\Pi_+-\Pi &= \gamma\beta(Y_+-V_+), && \text{(15f)}
\end{aligned}$$

The assumption $Z_+-Z\to 0$ implies that the left- and right-hand sides above all go to zero. Now we add subscript k to all variables $X,Y,\cdots$, and replacing X_+ by X_{k+1}, Y_+ by $Y_{k+1},\cdots$ and so on. Letting k go to infinity and noting $X_{k+1}=X_k+(X_{k+1}-X_k),\cdots$ and so on, where the second term vanishes asymptotically, we have

$$\begin{aligned}
(X_kY_k-M)Y_k^T+\Lambda_k &\to 0, && \text{(16a)}\\
X_k^T(X_kY_k-M)+\Pi_k &\to 0, && \text{(16b)}\\
\mathcal{P}_+(X_k+\Lambda_k/\alpha)-U_k &\to 0, && \text{(16c)}\\
\mathcal{P}_+(Y_k+\Pi_k/\beta)-V_k &\to 0, && \text{(16d)}\\
X_k-U_k &\to 0, && \text{(16e)}\\
Y_k-V_k &\to 0, && \text{(16f)}
\end{aligned}$$

where the terms $\alpha(X_k-U_k)$ and $\beta(Y_k-V_k)$ have been eliminated from (16a) and (16b), respectively, by invoking (16e) and (16f). Clearly, the first four equations in the KKT conditions (13) for problem (10) are satisfied at any limit point

$$\hat{Z}=(\hat{X},\hat{Y},\hat{U},\hat{V},\hat{\Lambda},\hat{\Pi}).$$

The nonnegativity of $\hat{U}$ and $\hat{V}$ are guaranteed by the algorithm construction. Therefore, we only need to verify the non-positivity of $\hat{\Lambda}$ and $\hat{\Pi}$, and the complementarity between $\hat{U}$ and $\hat{\Lambda}$ and between $\hat{V}$ and $\hat{\Pi}$. Now we examine the following two equations derived from (16c) and (16d), respectively,

$$\begin{aligned}
\mathcal{P}_+(\hat{X}+\hat{\Lambda}/\alpha) &= \hat{U}, && \text{(17a)}\\
\mathcal{P}_+(\hat{Y}+\hat{\Pi}/\beta) &= \hat{V}. && \text{(17b)}
\end{aligned}$$

If $\hat{U}_{ij}=\hat{X}_{ij}=0$, then (17a) reduces to $\mathcal{P}_+(\hat{\Lambda}/\alpha)_{ij}=0$ yielding $(\hat{\Lambda})_{ij}\le 0$. On the other hand, If $\hat{U}_{ij}=\hat{X}_{ij}>0$, then (17a) implies that $\hat{\Lambda}_{ij}=0$. This proves the non-positivity of $\hat{\Lambda}$ and the complementarity between $\hat{U}$ and $\hat{\Lambda}$. The same argument can be applied to (17b), due to the identical structure, to prove the non-positivity of $\hat{\Pi}$ and the complementarity between $\hat{V}$ and $\hat{\Pi}$.

We have verified the statement concerning the sequence $\{Z_k\}_{k=1}^{\infty}$ and problem (10). The statement concerning the sequence $\{(X_k,Y_k)\}_{k=1}^{\infty}$ and problem (1) follows directly from the equivalence between the two problems. This complete the proof. □

The following corollary follows immediately.

Corollary 1. *Whenever $\{Z_k\}_{k=1}^{\infty}$ converges, it converges to a KKT point*

Far from being satisfactory, the above simple result nevertheless provides some assurance on the behavior of the ADM algorithm applied to the non-convex NMF problem. Further theoretical studies in this direction are certainly desirable.

4 Computational Results

We conducted a series of numerical experiments to compare the proposed ADM algorithm with the two algorithms ALS and Mult discussed in Section 1.2. In our opinion, these two algorithms still represent the state of the art in this area because (1) they appear to be the most widely used algorithms by far, judging from NMF application papers published up to this date; (2) they remain the only algorithms implemented in the Matlab Statistics Toolbox today. In addition, based on our limited observations, the performance of some newer algorithms (whose Matlab implementations were readily available) is not consistently and convincingly better than that of ALS or Mult when tested on a wide range of problem sizes and characteristics. To be more thorough, we also included some additional comparison with two selected recent algorithms (see Section 4.4) beside ALS and Mult.

4.1 Implementational and experimental details

A pseudo code for the implemented ADM algorithm is as follows.

```
**Input**: $M \in \mathbb{R}^{m\times n}$, integers $k, maxiter > 0$ and $tol > 0$
**Output**: $X \in \mathbb{R}^{m\times k}$ and $Y \in \mathbb{R}^{k\times n}$
Set $\alpha, \beta, \gamma > 0$, and $Y$ to a nonnegative random matrix.
Set $U, V, \Lambda, \Pi$ to zero matrices of appropriate sizes.
for $k = 1$, maxiter do
    Update $(X, Y, U, V, \Lambda, \Pi)$ by the formulas in (12) ;
    if a stopping criterion relative to tol is met then
        exit and output $X$ and $Y$;
    end
end
```

The most important algorithmic parameters are $\alpha, \beta, \gamma > 0$. In our implementation (which was still preliminary), we set $\gamma = 1.618$ and $\alpha = \beta$. Short of solid theoretical guidances, we used the following heuristic to select α: we first scale M so that $\|M\|_F =$ 5e+06, then set $\alpha = 2000m/k$. This selection seems to have worked well for the tested matrices. A stopping criterion is met, relative to a tolerance value tol, if one the the following 3 conditions is satisfied:

$$\begin{aligned} |f_k - f_{k+1}|/|f_k| &\leq tol \\ \|F_k\|_F/\|F_0\|_F &\leq tol \\ f_k &\leq tol \end{aligned}$$

where $f_k = \|X_kY_k - M\|_F^2/2$ is the objective value and F_k is the optimality residue $F(X_k, Y_k)$ where F is defined in (2). Moreover, we require that the first condition above must be satisfied at three consecutive iterations.

Throughout the experiments and for all tested algorithms, we set the maximum number of iterations to $maxiter = 500$ and the tolerance value to $tol =$ 1e-07 unless otherwise specified. We also use the same random initial guesses to start all algorithms tested. The random numbers are generated by the Matlab command `rand` that are uniformly distributed in the interval $[0, 1]$. Other than specifying $maxiter$ and tol, we always used the default settings of the tested algorithms.

All numerical experiments were run on a MacBook Pro laptop computer with an Intel Core 2 Duo processor at 2.8GHz with 4GB RAM under Matlab version 7.10 (R2010a Prerelease).

4.2 Tests on images

To visualize results, we applied the three NMF algorithms, ADM, ALS and Mult, to two grayscale images and compared the obtained NMF results with those of the plain SVD method, despite that natural image compression has not been considered as one of the strength areas of the NMF technique. The two grayscale images, Kittens on the left and Panda on the right in Figure 1, have resolutions 768×1024 and 1200×1600, respectively. We used the Matlab command `[U,S,V] = svds(M,k)` to calculate a unconstrained, rank-k optimal approximation to the matrix M, where the Matlab function `svds(M,k)` computes a rank-k partial SVD decomposition of M.

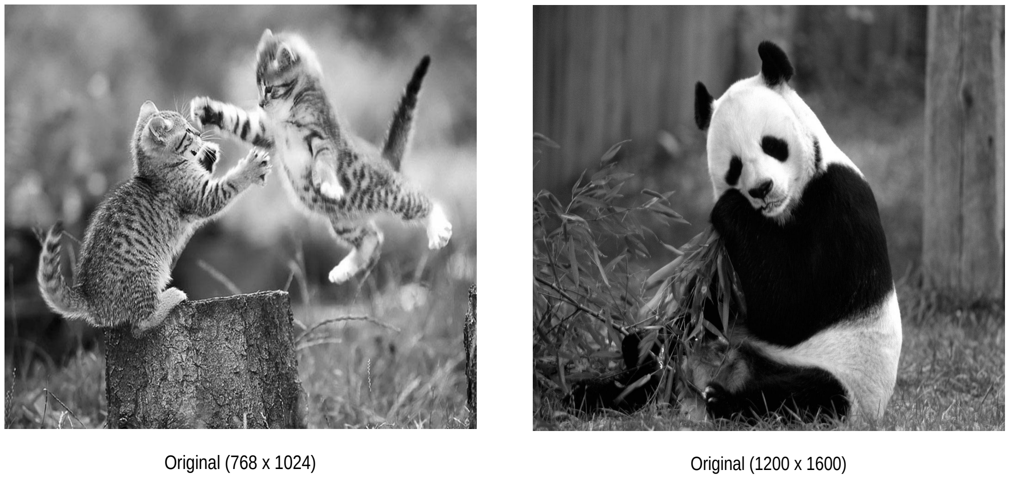


Figure 1: Original images: Kittens and Panda.

The results on image Kittens are given in Figure 2 for $k = 48$. We observe from Figure 2 that while SVD, ADM and Mult gave visually similar results with relative errors equal to 1.01e-01, 1.09e-01 and 1.16e-01, respectively, ALS produced a significantly inferior image that has a relative error at 2.34e-01. The relative errors are calculated by the formula:

$$\text{RelErr} \triangleq \frac{\|XY - M\|_F}{\|M\|_F} \tag{19}$$

where M represents the original image and XY is a low-rank approximation (in the SVD case $XY = USV^T$).

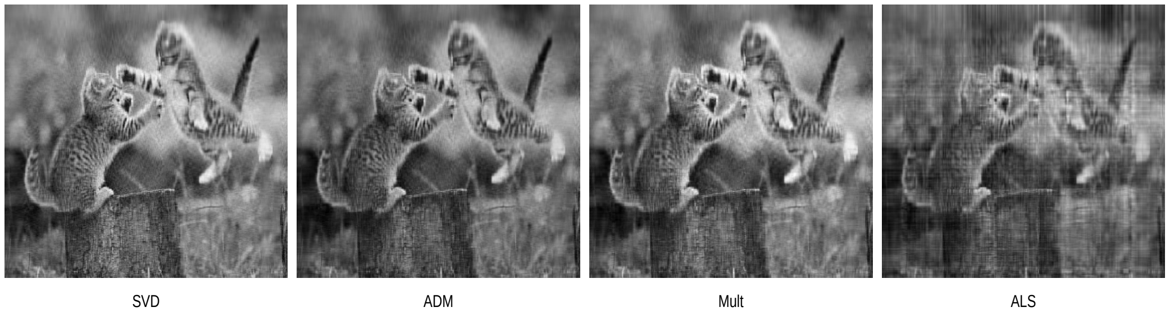


Figure 2: Results on image Kittens for $k = 48$.

On image Panda, ALS also performed much more poorly than the others. For example, for $k = 30$ the relative error and solution time for ALS were, respectively, 2.02e-01 and 3.30e+02 (in seconds), much worse than the corresponding numbers for the other methods (see Table 1). Due to its lack of competitiveness, we omit ALS form the reported results on image Panda.

Table 1: Relative error and solution time (in seconds) on image Panda

Method	SVD		ADM		Mult	
Rank	Rel. Error	Time	Rel. Error	Time	Rel. Error	Time
$k = 120$	4.76e-02	1.32e+01	6.04e-02	4.97e+01	7.23e-02	1.03e+02
$k = 60$	7.53e-02	6.85e+00	8.97e-02	2.35e+01	9.71e-02	6.20e+01
$k = 30$	1.11e-01	4.74e+00	1.23e-01	1.61e+01	1.27e-01	6.16e+01
$k = 15$	1.49e-01	3.07e+00	1.59e-01	1.09e+01	1.62e-01	5.17e+01

The results on image Panda are given in Table 1 and Figure 3 for the three methods, SVD, ADM and Mult. Table 1 show that the unconstrained SVD obtained slightly better relative errors, as expected, than those of ADM which in turn are slightly better than those of Mult in all tested cases. In terms of computing times, ADM is about over three times more expensive than SVD, while Mult is 2-5 times more expensive than ADM. However, it is interesting to note from Figure 3 that even though the SVD results have smaller relative errors, they are visually less appealing than the NMF results when $k = 30$ and 15 because of more severe loss of black-and-white contrast. Apparently, this loss of contrast was due to the fact at SVD allows negative entries in its low-rank approximations.

4.3 Tests on random matrices

We did rather extensive tests on random matrices that are generated as follows. A rank-r nonnegative test matrix $M \in \mathbb{R}^{m\times n}$ has the form $M = LDR$ where $L \in \mathbb{R}^{m\times r}$ and $R \in \mathbb{R}^{r\times n}$ are nonnegative and randomly generated using the Matlab command `rand`, while D is an $r \times r$ positive diagonal matrix with $(1, 2, \cdots, r)$ on its diagonal. This diagonal scaling makes M slightly to moderately ill-conditioned depending on how large r is.

Numerical results on three sets of random matrices are given in Figure 4 where the left column is for approximation quality and the right for computing times. In the first test set, $m = n = r = 500$ so the random matrices are of full rank. We ran the three algorithms, ADM, ALS and Mult, for $k = 50, 100, 150, \cdots, 500$. In the second test, again $m = n = 500$, but r varied from 10 to 100 with increment 10, while $k = r/2$. In the third test, we set $r = k = 30$ and varied $m = n$ from 100 to 1000 with increment 100. Each test set is corresponding to a row in Figure 4 where the obtained relative errors and computing times are plotted in the left and right figures of each row, respectively. The relative errors are defined as in (19) and the computing time was measured in seconds.

It should be evident from Figure 4 that in all the tests the ADM algorithm clearly outperformed both the ALS and Mult algorithms. It not only obtained the best quality in approximation but did so with the fastest time in all the tested cases without exception. In the first two test sets, the gap in solution quality widens as the approximation rank k increases, while the gap in solution time slightly narrows. In the third test set where r and k are equal and fixed, the quality gap narrows as problem sizes increase while the gap in solution time widens.

It is interesting to note from the third row of Figure 4 that when $k = r$ the algorithms seemed to have reached proximity of global minima since the final relative residue values were close to zero. This is particular true for the ADM algorithm when the problems were relatively small, but if more iterations were allowed, higher accuracy could be attained for larger problems as well. Indeed, our more extensive tests have shown that on matrices that are products of random low-rank nonnegative matrices, the ADM algorithm appears to be able to reach global optimality at high probability, which approached the unity at least in some test cases.

We also tested other types of random matrices and observed similar advantages of the ADM algorithm over the other two algorithms. For example, we tested on matrices that are products of nonnegative, low-rank, sparse matrices generated by using the Matlab command `sprandn` and then taking absolute values. We did observe improved performance of the ALS algorithm on such sparse problems, especially in cases where $k = r$ and global minima were reachable at high probability (in those cases, ALS can sometimes attain better accuracy than AMD). In Section 4.5, we will present test results on the extreme case where data matrices are positive diagonal so that the global minima of the NMF problems are always known.

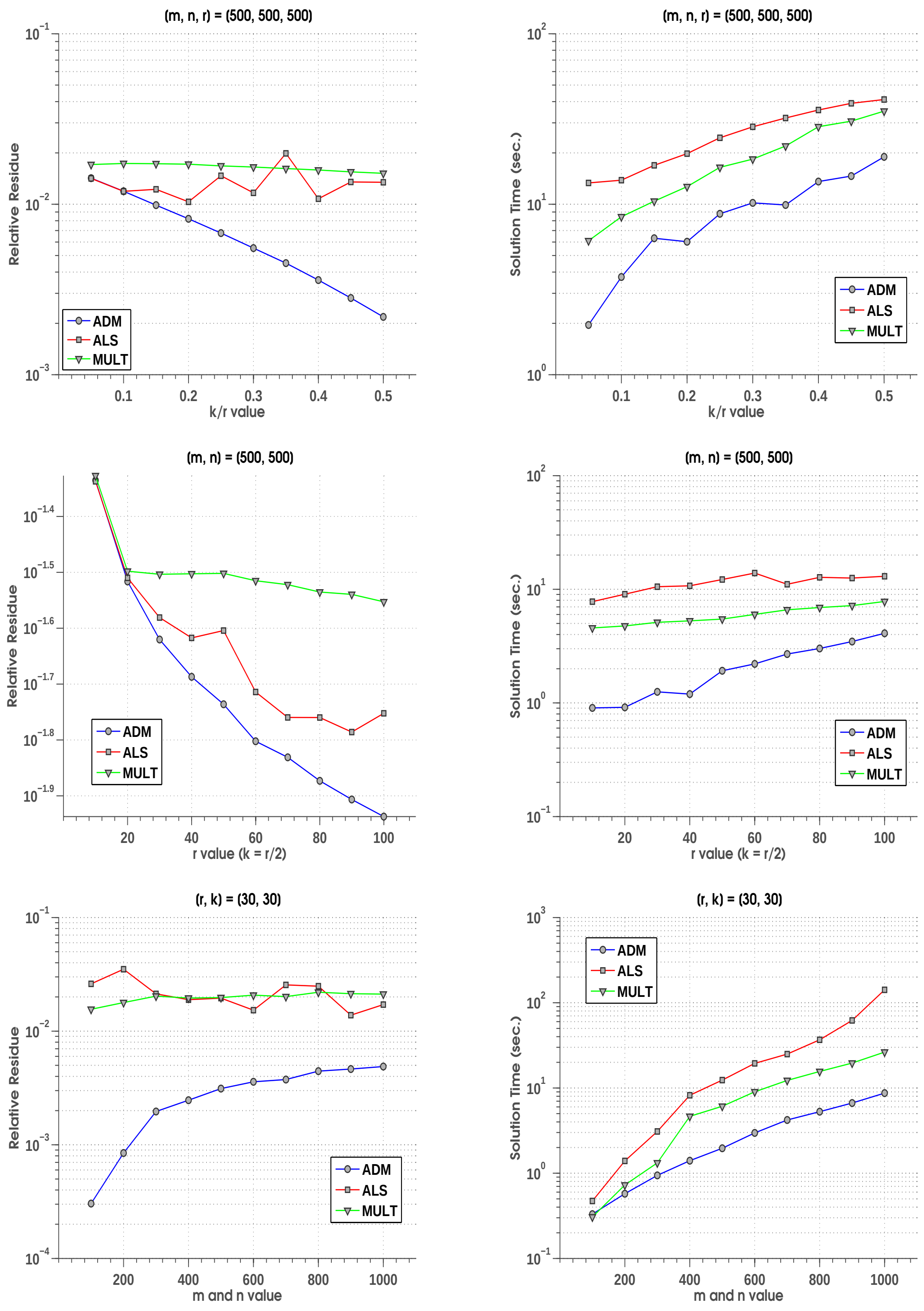


Figure 4: Comparison of the 3 algorithms on random test problems.

4.4 Comparison to RRI and Gradient Projection

A considerable number of algorithms have recently been proposed for solving the NMF problem. In this subsection, we compare the ADM algorithm briefly with two of the recent algorithms for which Matlab codes were readily available online. One is the rank-1 residue iterations (RRI) algorithm proposed by Ho, Van Dooren and Blondel [10] (also independently in [2]). RRI successively updates each rank-1 term in the sum $XY = \sum_{j=1}^{k} x_j y_j^T$ while keeping the other $k-1$ terms fixed, where x_j is the jth column of X and y_j^T the jth row of Y. Another is a projected gradient method proposed by Lin [14]. We selected RRI because it was shown in [2] to have outperformed a number of other algorithms including ALS and Mult. Beside its ready availability, the projected gradient method of Lin [14] was selected as a representative of algorithms based on the general principle of gradient descent. There are certainly other algorithms that deserve to be included for comparison (for example, those in [8, 11] among many candidates), but since our comparison was not meant to be exhaustive, we only selected the RRI algorithm and Lin's algorithm for the interest of space. For both of these algorithms, we used Matlab codes provided by the authors to do comparison. Results obtained from this comparison is given in Figure 5.

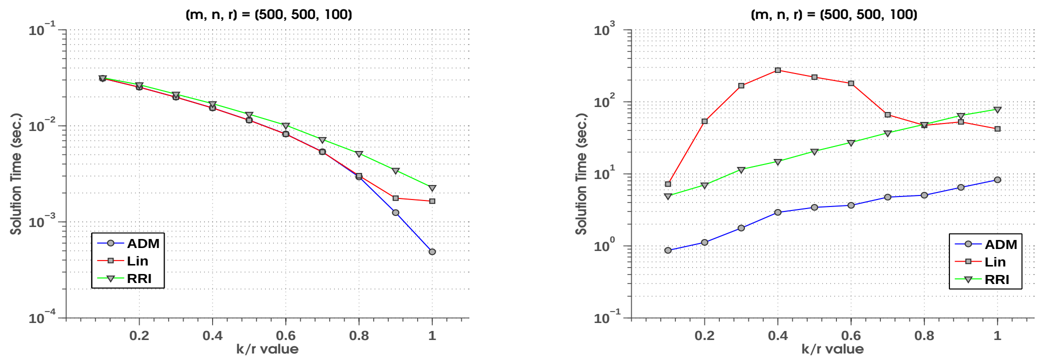


Figure 5: Comparison of ADM, Lin and RRI Algorithms on random matrices.

In these experiments, we used the same construction of random matrices as described in the previous subsection. The matrices are of 500×500 with rank 100. The low rank value k was varied from 10 to 100 with increment 10. The same random initial guesses and maximum iteration number, $maxiter = 500$, were used for all three algorithms. The stopping tolerance for ADM was 1.0e-07, and for Lin's algorithm it was 1.0e-06 (since 1.0e-07 was too restrictive for Lin's code on this test). On the othet hand, RRI does not require a tolerance.

As is evident in Figure 5, in comparison to the other algorithms, ADM obtained either equally good or better solution quality with much shorter computing times. Also, on these problems Lin's algorithm obtained better quality than RRI but generally required longer computing times.

4.5 Chances of getting global optima

Due to non-convexity, the globally optimal function value for an NMF problem is generally unknown. To obtain indications on the ability of NMF algorithms to reach global optimality, we test these algorithms on simple positive diagonal matrices for which global optima are known. Specifically, we construct test matrices $M \in \mathbb{R}^{n\times n}$ of the form

$$M = 10I + tD_k$$

where $t > 0$ is a parameter, and D_k is a diagonal matrix with k ones and $n-k$ zeros on its diagonal, while the positions of the ones in D_k are randomized. As such, M has k diagonal elements taking the value $10+t$ and $n-k$ taking 10. It is clear that any globally optimal rank-k nonnegative matrix factorization $\bar{X}\bar{Y}$ is a diagonal matrix that retains the k largest diagonal elements of M at $10+t$ and zeros out the other $n-k$ diagonal elements at 10. For example, if the first k diagonal elements of M are $10+t$, then a globally optimal rank-k nonnegative matrix factorization consists of

$$\bar{X} = (10+t)\left[\begin{array}{c} I_{k\times k} \\ 0 \end{array}\right] \quad \text{and} \quad \bar{Y} = \left[\begin{array}{cc} I_{k\times k} & 0 \end{array}\right],$$

with the global minimum objective value

$$f(\bar{X},\bar{Y}) = \|\bar{X}\bar{Y} - M\|_F^2/2 = (n-k)100/2.$$

For $t = 1, 2, \cdots, 20$, we run each algorithm on 100 random instances and record the number of runs succeeding in reaching global optima within a prescribed accuracy. Specifically, a run is regarded as successful if the returned solution (X, Y) satisfies the criterion

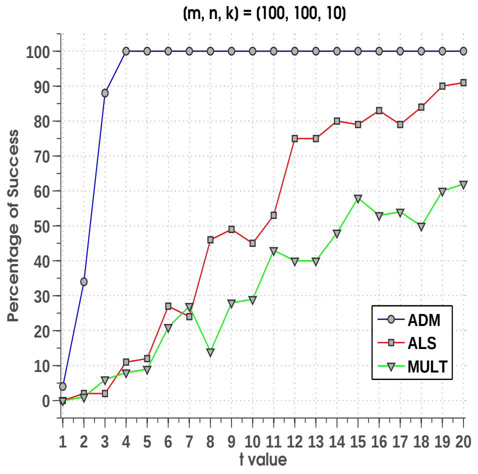
$$\frac{\|XY - M\|_F^2 - 100(n-k)}{100(n-k)} \le 10^{-2}(t/10), \tag{20}$$

that is, the relative error in objective as compared to a global optimum is within one percent times $t/10$. The accuracy requirement is scaled by the factor $t/10$ because of the following reason. Let XY be diagonal and of rank k with $k-1$ diagonal elements coinciding with those of M at the value of $10+t$ and another one equal to 10. Then the residue value at such a pair (X, Y) is

$$\|XY - M\|_F^2 = (n-k-1)10^2 + (10+t)^2 = 100(n-k) + t(t+20),$$

corresponding to a left-hand side in (20) at $[t(t+20)/(n-k)]10^{-2}$ which is less than 10^{-2} when t and k are relatively small, say for $t \le 3$ and $k \le 30$. The scaling factor $t/10$ precludes such kind of (X, Y) from being accepted as good approximations for small t values.

Out of many instances of such experiments, results for two typical runs were presented in Figure 6 where rates of successful runs in approximating global optima, as defined in (20), were plotted for the three algorithms and for $t = 1, 2, \cdots, 20$. In both instances, the matrix sizes are $m = n = 100$. As can be seen from the picture on the left, for $k = 10$ the rates of success generally increase as t increases. Although the performance of the ADM algorithm dominated the other two algorithms for $k = 10$, its behavior became rather counter-intuitive when $k = 15$ as can be seen from the picture on the right where the success rate drops as t passes 7.



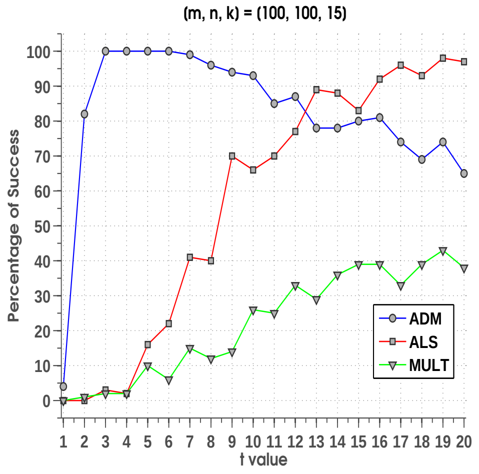


Figure 6: Success rates of global optimization for the 3 algorithms on diagonal test problems. For each t value and each algorithm, 100 random trials were performed.

Since the diagonal test matrices are not representative of any meaningful data, results from such experiments may not be indicative of any strength or weakness of the tested algorithms. However, these results do suggest that (1) under favorable conditions chances of getting global optima could potentially be quite high, particularly for the ADM algorithm; and (2) algorithms may exhibit unexpected behavior under different conditions in terms of approximating global optima. Further studies are needed to better understand issues related to global optimization.

5 Concluding Remarks

With auxiliary variables and Lagrangian multipliers, and requiring setting a couple of parameters, the ADM algorithm is slightly more involved than both the ALS and the Mult algorithms. Nevertheless, it is still quite simple. More importantly, in our tests it has consistently and notably outperformed all the peers in comparison, measured either in solution quality or speed, or in both aspects. The per-iteration complexity of the ADM algorithm is mainly two matrix-vector multiplications when the approximation rank k is relatively small (as is the case in most applications).

The ADM approach can be applied to other NMF-related models such as weighted NMF problems where weighted Frobenius norms are used, semi-NMF problems where only one of the two factors is required to be nonnegative, and various regularized models such as ℓ_1-norm regularization for sparse factorizations. It is reasonable to predict that the resulting algorithms should be competitive as is the case for the plain NMF problem.

Although from an empirical point of view the ADM approach applied to non-convex problems appears to have robust convergence properties when penalty parameters are appropriately chosen, from a theoretical point of view these properties remain poorly understood. An important issue concerns the likelihood of convergence to global optima. For some classes of problems, convergence to global optima does appear to be highly probable despite non-convexity. These topics should be of interest for further studies.

References

[1] M. Berry, M. Brownea, A. Langvilleb, V. Paucac and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. Computational Statistics and Data Analysis. Vol. 52, Issue 1, 15 September 2007, pp. 155-173.

[2] A. Cichocki, R. Zdunek, and S. Amari. Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization. In Proceedings of Independent Component Analysis, ICA 2007, Lon- don, UK, September 9-12, 2007, Lecture Notes in Computer Science, Springer, 4666:169-176, 2007.

[3] A. Cichocki, M. Morup, P. Smaragdis, W. Wang, and R. Zdunek (eds). *Advances in Nonnegative Matrix and Tensor Factorization.* Computational Intelligence and Neuroscience. Volume 2008. Hindawi Publishing Corporation. `http://www.hindawi.com/`.

[4] A. Cichocki, R. Zdunek, A.-H. Phan and S. Amari. *Nonnegative Matrix and Tensor Factorizations – Applications to Exploratory Multiway Data Analysis and Blind Source Separation.* 2009. John Wiley & Sons, Ltd. The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom.

[5] Fortin, M. and Glowinski, R. [*Augmented Lagrangian Methods.* North-Holland, Amsterdam, New York (1983).

[6] D. Gabay, and B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. Comp. Math. Appl., vol. 2, pp. 17-40, 1976.

[7] R. Glowinski, and A. Marrocco. *Sur lapproximation par elements finis dordre un, et la resolution par penalisation-dualite dune classe de problemes de Dirichlet nonlineaires*, Rev. Francaise dAut. Inf. Rech. Oper., R-2, pp. 41-76, 1975.

[8] Gonzalez, E. and Zhang, Y. Accelerating the Lee-Seung algorithm for nonnegative matrix factorization. CAAM Technical Report TR05-02, Rice University. March 2005.

[9] M. R. Hestenes, *Multiplier and gradient methods*, J. Optimiz. Theory App., vol. 4, pp. 303–320, 1969.

[10] N.-D. Ho, Paul Van Dooren, Vincent D. Blondel. Descent methods for Nonnegative Matrix Factorization arXiv:0801.3199v3 [cs.NA], 24 Aug 2009.

[11] H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM Journal on Matrix Analysis and Applications 30 (2): 713-730. 2008.

[12] Lee, D. and Seung, H. Learning the Parts of Objects by Non-Negative Matrix Factorization. Nature 401, 788-791. 1999.

[13] Lee, D. and Seung, H. Algorithms for Non-Negative Matrix Factorization. Advances in Neural Information Processing Systems 13, 556-562. 2001.

[14] C.-J. Lin Projected Gradient Methods for Nonnegative Matrix Factorization. Neural Computing. October 2007, Vol. 19, No. 10, Pages 2756-2779

[15] Paatero, P. and Tapper, U. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics 5, 111-126. 1994.

[16] P. Paatero. Least squares formulation of robust non-negative factor analysis. Chemometrics and Intelligent Laboratory Systems 37, pp. 23–35. 1997.

[17] P. Paatero. The Multilinear Engine: A Table-Driven, Least Squares Program for Solving Multilinear Problems, including the n-Way Parallel Factor Analysis Model. Journal of Computational and Graphical Statistics 8 (4): 854-888, 1999.

[18] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, New York, NY, pp. 283–298, 1969.

[19] R. T. Rockafellar, *The multiplier method of Hestenes and Powell applied to convex programming*, J. Optimiz. Theory App., vol. 12, pp. 555–562, 1973.

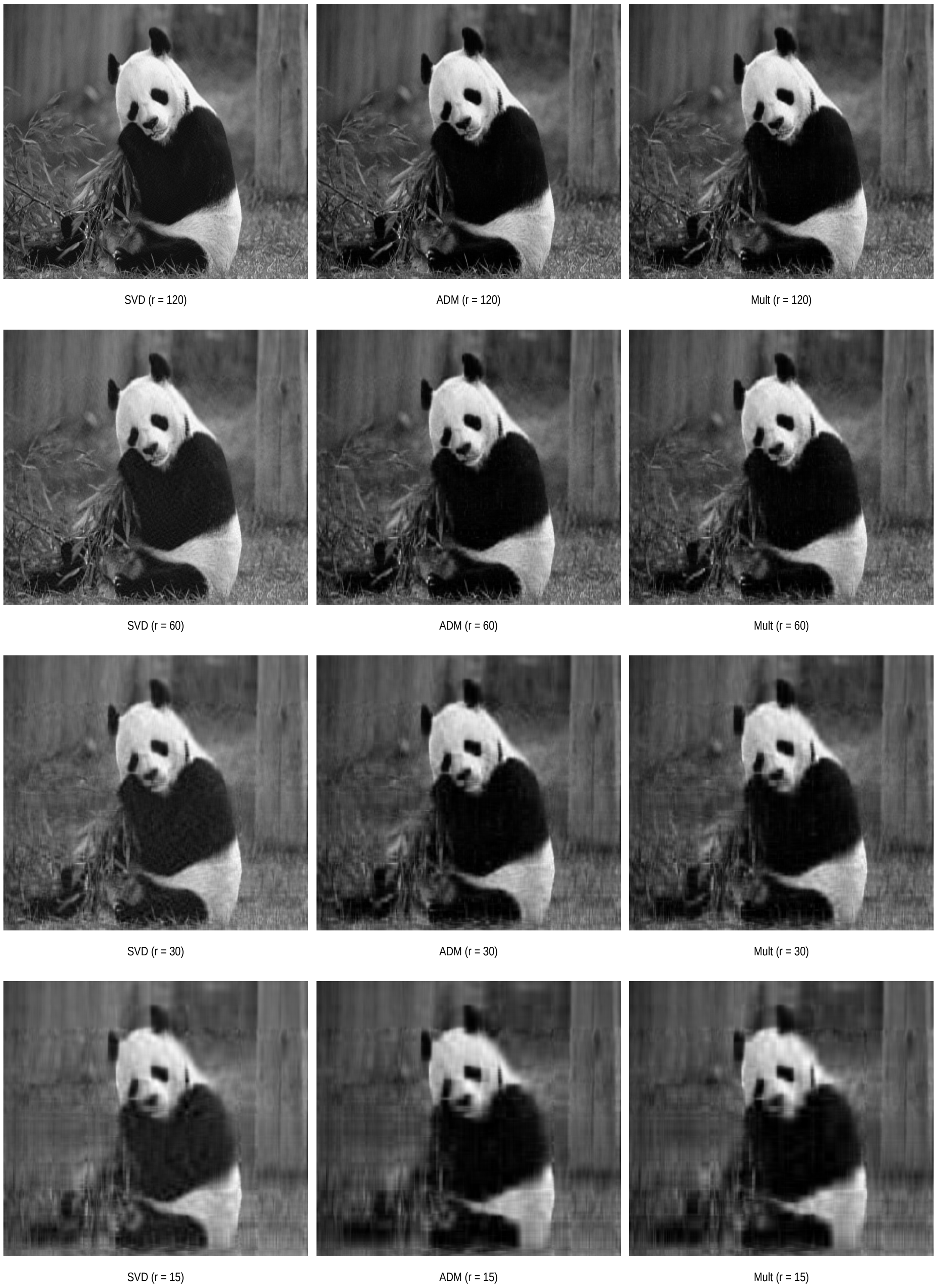


Figure 3: Results on the 1200×1600 image Panda for rank $k = 120, 60, 30, 15$ (top to bottom) and for algorithms SVD, ADM and Mult (left to right).