Nonnegative Matrix Factoriaztion

*Report 3 on the course "Numerical Optimization".

1st Chen Yihang

Peking University

1700010780

Abstract

In this report, I give a brief review and implementation of existing nonnegative matrix factoriaztion method. Numerical experiments are performed on synthetic and real-life datasets.

CONTENTS

I. Introduction

Given a non-negative matrix V, find non-negative matrix factors W and H such that:

$$V \approx WH \tag{I.1}$$

We formulate the problem as the following optimization problem:

$$\min \quad f(V, W, H), \quad \text{s.t.} \quad W, H \geq 0 \tag{I.2}$$

II. Multiplicative update rules

We briefly introduce multiplicative update rules [Lee and Seung, 2001]. This paper consider two alternative formulations of NMF, and utilize a rescaled gradient descent on these formulations. The step size is chosen such that the positiveness can be guaranteed every step, and the convergence can be rigorously proved.

1) $f(V, W, H) = \|V - WH\|_F^2$. Then a simple additive update rule via gradient descent can be written as

$$H_{a\mu} \leftarrow H_{a\mu} + \eta_{a\mu}(W^\top V - W^\top WH)_{a\mu} \tag{II.1}$$

If we take $\eta_{a\mu} = \frac{H_{a\mu}}{(W^\top WH)_{a\mu}}$, we can get

$$H_{a\mu} \leftarrow H_{a\mu}\frac{(W^\top V)_{a\mu}}{(W^\top WH)_{a\mu}}, \quad W_{ia} \leftarrow W_{ia}\frac{(VH^\top)_{ia}}{(WHH^\top)_{ia}} \tag{II.2}$$

2) $f(V, W, H) = D(V\|WH)$. We take the step size to be $\eta_{a\mu} = \frac{H_{a\mu}}{\sum_i W_{ia}}$, the multiplicative rule is

$$H_{a\mu} \leftarrow H_{a\mu}\frac{\sum_i W_{ia}V_{i\mu}/(WH)_{i\mu}}{\sum_k W_{ka}}, \quad W_{ia} \leftarrow W_{ia}\frac{\sum_\mu H_{a\mu}V_{i\mu}/(WH)_{i\mu}}{\sum_\mu H_{a\mu}} \tag{II.3}$$

In particular, it is straightforward to see that this multiplicative factor is unity when $V = WH$, so that perfect reconstruction is necessarily a fixed point of the update rules.

However, despite its theoretical soundness, its performance is subpar compared to another framework, i.e. formulating NMF as an alternating nonnegative least squares problem.

III. Nonnegative Least Sqaure Framework

For simplicity, we consider the following notations in this section:

$$\min_{H \geq 0} \|V - HW\|_F^2 \tag{III.1}$$

There are two groups of algorithms for the NLS problems. The first group consists of the gradient descent and the Newton-type methods that are modified to satisfy the nonnegativity constraints using a projection operator. The second group consists of the active-set and the active-set-like methods, in which zero and nonzero variables are explicitly kept track of and a system of linear equations is solved at each iteration.

A. Projected Gradient Method

1) Vanilla projected gradient method for NMF: Two projected gradient methods are proposed in [Lin, 2007]. In this paper, $f(V, W, H) = \|V - WH\|_F^2$. The author first propose an improved projected gradient method. For a constraint optimization problem

$$\begin{aligned} \min_{x} \quad & f(x) \\ & l_i \leq x_i \leq u_i \end{aligned} \tag{III.2}$$

The projected gradient method can be stated as

$$x^{k+1} = P[x^k - \alpha_k \nabla f(x^k)] \tag{III.3}$$

and the full algorithm is stated in Algorithm 3.

Algorithm 1 Projected gradient method with a back-tracking line search

1: Given $0 < \beta < 1$, $0 < \sigma < 1$. Initialize any feasible x^1. Set $\alpha_0 = 1$.
2: **for** $k = 1, 2, \ldots$ **do**
3: Assign $\alpha_k \leftarrow \alpha_{k-1}$.
4: **if** α_k satisfies

$$f(x^{k+1}) - f(x^k) \leq \sigma \nabla f(x^k)^\top (x^{k+1} - x^k) \tag{III.4}$$

then repeatedly increase it by $\alpha^k \leftarrow \alpha^k/\beta$ until either the equation is not satisfied or $x(\alpha_k/\beta) = x(\alpha_k)$.
5: **else**repeatedly decrease α_k by $\alpha_k \leftarrow \alpha_k \beta$ until α_k satisfies III.4.
6: **end if**
7: Update x^k according to III.3.
8: **end for**

a) Alternating Non-negative Least Squares Using Projected Gradient Methods: We focus on solving

$$\begin{aligned} \min_{H} \quad & \tilde{f}(H) := \frac{1}{2}\|V - WH\|_F^2 \\ \text{s.t.} \quad & H \geq 0 \end{aligned} \tag{III.5}$$

Since $\tilde{f}$ is quadratic, the equation III.4 becomes

$$(1-\sigma)\left\langle \nabla \tilde{f}(H^k), H^{k+1} - H^k \right\rangle + \frac{1}{2}\left\langle H^{k+1} - H^k, (W^\top W)(H^{k+1} - H^k) \right\rangle \leq 0 \tag{III.6}$$

which is more computationally efficient. Similarily, we can define

$$\bar{f}(W) := \frac{1}{2}\|V^\top - H^\top W^\top\|_F^2$$

and use the identical procedures above.

b) Directly Applying Projected Gradients to NMF: We simultaneously update both matrices (W, H) by

$$(W, H) \leftarrow P[(W, H) - \alpha(\nabla_W f(W, H), \nabla_H f(W, H))] \tag{III.7}$$

and Algorithm 3 is applied. Note that since f is not quadratic, we cannot use the trick III.6 to accelerate the algorithm.

The detailed experiments settings are identical to the paper [Lin, 2007].

2) Barzilai-Borwein method for NMF: A key characteristic of the Barzilai-Borwein method in unconstrained quadratic programming is that the step-length is chosen by a closed-form formula without having to perform a line search:

$$\begin{aligned} x^{(i+1)} \leftarrow \left[x^{(i)} - \alpha^{(i)} \nabla g(x^{(i)})\right]_+ \quad & \text{with } \alpha^{(i)} = \frac{s^T s}{y^T s}, \quad \text{where} \\ s = x^{(i)} - x^{(i-1)} \quad & \text{and } \; y = \nabla g(x^{(i)}) - \nabla g(x^{(i-1)}) \end{aligned} \tag{III.8}$$

When the nonnegativity constraints are given, however, back-tracking line search still had to be employed. The nonmonotone line search method proposed in [Raydan, 1997] was modified to accommodate the NMF problem. The key difference is that, after an update $\tilde{x}_{k+1} = x_k - \alpha_k g_k$, where α_k is the Barzilai-Borwein step size, we need first project $\tilde{x}_{k+1}$ onto nonnegative vectors, and then perform nonmonotone line search. Several other

variants are proposed in [Han et al., 2010], yet the core idea is the same.

3) Quasi-Newton method for NMF: Kim et al. [Sra and Dhillon, 2007] proposed a quasi-Newton method by utilizing the second order information to improve convergence.

Adopting their notiations, we aim to solve the sub-problem

$$\min_{x\geq 0} \frac{1}{2}\|Gx - h\|_F^2$$

They partition the variables x into two groups, namely the free and fixed variables. The fixed variables are the components of x with active constraints (equality satisfied) that have a corresponding positive derivative. We index them by the fixed set, i.e.

$$I_+ = \{i | x_i = 0, [\nabla f(x)]_i > 0\} \tag{III.9}$$

Hence, the update can be written as

$$x^{i+1} \leftarrow \begin{pmatrix} [y^i - \alpha^i D^i \nabla g(y^i)]_+ \\ 0 \end{pmatrix}, \tag{III.10}$$

where y^i is a subvector of x^i with elements that are not optimal in terms of the Karush–Kuhn–Tucker (KKT) conditions, i.e. not in fixed set I_+. They efficiently updated D^i using the BFGS method and selected α^i by a back-tracking line search. Whereas Lin considered a stacked-up problem as in [Lin, 2007], the quasi-Newton method by Kim et al. [Sra and Dhillon, 2007] was applied to each column separately. This algorithm is implemented in "nnls_pgn.m". If we apply "nnls_pgn.m" directly, the performace is not satisfactory.

However, their description of exact version of NMF is quite unclear. It is quite confusing how they update the D during Step 3.6 in Algorithm 1. Besides, the stopping condition for the inner loop is also unclear, and I cannot find the authors' implementation to elucidate these points. Hence, I choose to implement the inexact version of quasi-Newton method, i.e. Algorithm 2, FNMA[I].

Details of the algorithm will be presented in the codes but will not be repeatedly stated here. But there are one main difference, I do not adopt its heuristics to choose α. Instead, I tune this parameter by hand.

Algorithm 2 FNMA$^{\mathrm{I}}$

Input: $\boldsymbol{A} \in \mathbb{R}_{+}^{M \times N}, \quad K, \tau \in \mathbb{N}, \alpha \in \mathbb{R}_{+}$.
Output: $\boldsymbol{B} \in \mathbb{R}_{+}^{M \times K}, \boldsymbol{C} \in \mathbb{R}_{+}^{K \times N}$
1. Initialize $\boldsymbol{B}^0, \boldsymbol{C}^0, t = 0$.
repeat
  2. $\boldsymbol{B} \leftarrow \boldsymbol{B}^t, \quad \boldsymbol{C}^{\text{old}} \leftarrow \boldsymbol{C}^t$.
  for $i = 1$ to τ **do**
    3.1. Compute the gradient matrix $\nabla_{\boldsymbol{C}} \mathcal{F}(\boldsymbol{B}; \boldsymbol{C}^{\text{old}})$.
    3.2. Compute fixed set I_+ for $\boldsymbol{C}^{\text{old}}$.
    3.3. Update $\boldsymbol{C}^{\text{old}}$ as:

$$\boldsymbol{U} \leftarrow \mathcal{Z}_+\left[\nabla_{\boldsymbol{C}} \mathcal{F}(\boldsymbol{B}; \boldsymbol{C}^{\text{old}})\right]; \quad \boldsymbol{U} \leftarrow \mathcal{Z}_+\left[\left(\boldsymbol{B}^T \boldsymbol{B}\right)^{-1} \boldsymbol{U}\right];$$
$$\boldsymbol{C}^{\text{new}} \leftarrow \mathcal{P}_+\left[\boldsymbol{C}^{\text{old}} - \alpha \boldsymbol{U}\right].$$

    3.4. $\boldsymbol{C}^{\text{old}} \leftarrow \boldsymbol{C}^{\text{new}}$.
  end for
  4. $\boldsymbol{C}^{t+1} \leftarrow \boldsymbol{C}^{\text{old}}$.
  5. $\boldsymbol{C} \leftarrow \boldsymbol{C}^{t+1}, \quad \boldsymbol{B}^{\text{old}} \leftarrow \boldsymbol{B}^k$.
  for $i = 1$ to τ **do**
    6.1. Compute the gradient matrix $\nabla_{\boldsymbol{B}} \mathcal{F}(\boldsymbol{B}^{\text{old}}; \boldsymbol{C})$.
    6.2. Compute fixed set I_+ for $\boldsymbol{B}^{\text{old}}$.
    6.3. Update $\boldsymbol{B}^{\text{old}}$ as:

$$\boldsymbol{U} \leftarrow \mathcal{Z}_+\left[\nabla_{\boldsymbol{B}} \mathcal{F}(\boldsymbol{B}^{old}; \boldsymbol{C})\right]; \quad \boldsymbol{U} \leftarrow \mathcal{Z}_+\left[\boldsymbol{U}\left(\boldsymbol{C}\boldsymbol{C}^T\right)^{-1}\right];$$
$$\boldsymbol{B}^{\text{new}} \leftarrow \mathcal{P}_+\left[\boldsymbol{B}^{\text{old}} - \alpha \boldsymbol{U}\right].$$

    6.4. $\boldsymbol{B}^{old} \leftarrow \boldsymbol{B}^{new}$.
  end for
  7. $\boldsymbol{B}^{t+1} \leftarrow \boldsymbol{B}^{old}$.
  8. $t \leftarrow t + 1$.
until Stopping criteria are met

B. Conjugate Gradient method - Proposed method

I tried to use the conjugate gradient method to solve the subproblem, since it has been proven effective in solving the unconstrained version. I apply the generalized conjugate gradient method [Liu and Storey, 1991] on this problem, and test it on various datasets. We find that this idea is not effective, since it is consuming in each iteration. Anyway, I write it in "nmf_cg.m".

C. Active Set method

The active-set method for the NLS problems is due to Lawson and Hanson [64]. A key observation is that, if the zero and nonzero elements of the final solution are known in advance, the solution can be easily computed by solving an unconstrained least squares

problem for the nonzero variables and setting the rest to zeros. The sets of zero and nonzero variables are referred to as active and passive sets, respectively. In the active-set method, so-called workings sets are kept track of until the optimal active and passive sets are found. A rough pseudo-code for the active-set method is shown in Algorithm 1.

Algorithm 2 Outline for Active-set method for $\min_{x\geq 0} f(x) = \|Ax - b\|_2^2$

1: Initialize x.
2: Set $\mathcal{I}, \mathcal{E}$ to be indices representing zero and nonzero variables. Let $x_{\mathcal{I}}$ and $x_{\mathcal{E}}$ denote the subvectors of x with corresponding indices, and $A_{\mathcal{I}}$ and $A_{\mathcal{E}}$ denote the submatrices of B with corresponding indices.
3: **for** $i = 1, 2, \cdots$ **do**
4: Solve the unconstrained least squares problem

$$\min_{x\geq 0} \|A_{\mathcal{E}} z - b\|_2^2. \tag{III.11}$$

5: Check if the solution is nonn and satisfies KKT condition. If so, set $x_{\mathcal{E}} \leftarrow z$, set $x_{\mathcal{I}}$ with zeros, and return x. Otherwise, update $x, \mathcal{I}, \mathcal{E}$.
6: **end for**

The active-set methods possess a property that the objective function decreases after each iteration; however, maintaining this property often limits its scalability. A main computational burden of the active-set methods is in solving the unconstrained least squares problem; hence, the number of iterations required until termination considerably affects the computation cost. In order to achieve the monotonic decreasing property, typically only one variable is exchanged between working sets per iteration. As a result, when the number of unknowns is large, the number of iterations required for termination grows, slowing down the method. The block principal pivoting method developed by Kim and Park [Kim and Park, 2011] allows the exchanges of multiple variables between working sets. This method does not maintain the nonnegativity of intermediate vectors nor the monotonic decrease of the objective function, but it requires a smaller number of iterations until termination than the active set methods. It is worth emphasizing that the grouping-based speed-up technique, which was earlier devised for the active-set method, is also effective with the block principal pivoting method for the NMF computation.

a) NNLS with a single right-hand side vector: For $\min_{x\geq 0} f(x) = \|Ax - b\|_2^2$, the KKT condition can be written as

$$y = A^\top Ax - A^\top b, \quad y \geq 0, x \geq 0, \quad x_i y_i = 0 \tag{III.12}$$

For subset F, G of $[n]$, initially, we assign $x_G = 0, y_F = 0$. The complementary slackness is always satisfied, we can update x_F, y_G via

$$x_F = \arg\min_{x_F} \|A_F x_F - b\|_2^2 \tag{III.13a}$$

$$y_G = A_G^\top (A_F x_F - b) \tag{III.13b}$$

If x, y is infeasible, i.e., $x_F \geq 0$ and $y_G \geq 0$ does not hold. Then, we can define V as all the infeasible indices

$$V = \{i \in F : x_i < 0\} \cup \{i \in G : y_i < 0\} \tag{III.14}$$

and choose a subset of $\tilde{V} \subset V$ and the F, G is updated by

$$F = (F - \tilde{V}) \cup (\tilde{V} \cap G), \quad G = (G - \tilde{V}) \cup (\tilde{V} \cap F) \tag{III.15}$$

Algorithm 3 Block principal pivoting method for the NNLS problem with a single right-hand side vector.

Initialize $F = \emptyset, G = [n], x = 0, y = -A^\top b. \alpha = 3, \beta = n + 1$.
Update x_F and y_G
while x_F and y_G is infeasible **do**
    Compute the infeasible indices set V.
    if $|V| < \beta$ **then**
        Set $\beta = |V|, \alpha = 3, \tilde{V} = V$.
    else if $|V| \geq \beta$ and $\alpha \geq 1$ **then**
        Set $\alpha = \alpha - 1, \tilde{V} = V$.
    else if $|V| \geq \beta$ and $\alpha = 0$ **then**
        Set $\tilde{V}$ by $\tilde{V} = \{i : i = \max i \in V\}$.
    end if
    Update F, G. Update x_F, y_G.
end while

In order to speed up the search procedure, one usually uses $\tilde{V} = V$, which we call the full exchange rule. The full exchange rule means that we exchange all variables of

F and G that do not satisfy the KKT condition, and the rule accelerates computation by reducing the number of iterations required until termination. However, contrary to the active set method in which the variable to exchange is carefully selected to reduce the objective function, the full exchange rule may lead to a cycle and fail to find an optimal solution, though rarely occurs. To ensure finite termination, where only the infeasible variable with the largest index is exchanged, is a single principal pivoting rule.

b) NNLS with multiple right-hand side vectors: Now suppose we need to solve the following NNLS problem:

$$\min_{X \geq 0} \|AX - B\|_F^2 \tag{III.16}$$

The process of the single right-hand side vectors are required to perform individually on every column of B.

D. Comparison

A main difference between the projected iterative methods and the active-set-like methods for the NLS problems lies in their convergence or termination. In projected iterative methods, a sequence of tentative solutions is generated so that an optimal solution is approached in the limit. In practice, one has to somehow stop iterations and return the current estimate, which might be only an approximation of the solution. In the active-set and active-set-like methods, in contrast, there is no concept of a limit point. Tentative solutions are generated with a goal of finding the optimal active and passive set partitioning, which is guaranteed to be found in a finite number of iterations since there are only a finite number of possible active and passive set partitionings. Once the optimal active and passive sets are found, the methods terminate. There are trade-offs of these behavior. While the projected iterative methods may return an approximate solution after a few number of iterations, the active-set and active-set-like methods only return a solution after they terminate. After the termination, however, the solution from the active-set-like methods is an exact solution only subject to numerical rounding errors while the solution from the projected iterative methods might be an approximate one.

IV. Numerical Results

A. Stopping codition

If we take the stopping condition as

$$f(W_{k-1}, H_{k-1}) - f(W_k - H_k) \leq \varepsilon \tag{IV.1}$$

the algorithm might be terminated in advance. Hence, we should use the first order optimality condition to formulate a more reasonable stopping condition.

The KKT condition of

$$\min_{W,H\geq 0} f(W,H) = \frac{1}{2}\|V - WH\|_F^2 \tag{IV.2}$$

can be written as

$$W \geq 0, H \geq 0 \tag{IV.3a}$$

$$\nabla f_W \geq 0, \nabla f_H \geq 0 \tag{IV.3b}$$

$$W \odot \nabla f_W = 0, H \odot \nabla f_H = 0 \tag{IV.3c}$$

If we define

$$(\nabla^p f_W)_{mk} := \begin{cases} (\nabla^p f_W)_{mk} & \text{if } (\nabla^p f_W)_{mk} < 0 \text{ or } W_{mk} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{IV.4}$$

$\nabla^p f_H$ is similarly defined. Then the condition IV.3 can be restated as

$$\nabla^p f_W = 0, \text{ and } \nabla^p f_H = 0. \tag{IV.5}$$

Hence, we take the stopping condition as

$$\| \left[\nabla_{W_k} f, (\nabla_{H_k} f)^\top\right] \|_F \leq \varepsilon \| \left[\nabla_{W_0} f, (\nabla_{H_0} f)^\top\right] \|_F \tag{IV.6}$$

In the following, we takt $\varepsilon = 10^{-4}$.

B. Code description

- nmf_mu.m: multiplicative weights method.

- nmf_pgd.m: projected gradient method with two variants. The sub-function "nmf_subpgd.m" are excerpted from [Lin, 2007].
- nmf_BB.m: Borwein-Borwein method.
- nmf_newton_inexact.m: FNMA$^{\mathrm{I}}$ in [Sra and Dhillon, 2007].
- nmf_anls.m: active set method with three variants. The sub-function "anls_bpp.m", 'anls_asgivens.m', 'anls_asgroup.m', are excerpted from the author's homepage [1].
- png.m: solves the nonnegative least squares problems via quasi-Newton method.
- mm_to_msm.m: data prepocessing for the "BBC Sports Datasets".
- display_image.m: display the face figures.
- test.m: performs numerical tests of all the algorithms.

C. Test

In the table, "rel_cost" means $\frac{\|V-WH\|_F}{\|V\|_F}$, and "rel_pgn" means

$$\| \left[\nabla_{W_k} f, (\nabla_{H_k} f)^\top\right] \|_F / \| \left[\nabla_{W_0} f, (\nabla_{H_0} f)^\top\right] \|_F$$

For the OCR, we take $r = 25$, and for the Yale, we take $r = 15$, otherwise, we take $r = 10$.

There are two main observation:

1) If we take all-ones matrices as initial value for PGD type methods. (active set methods cannot take all-ones matrices as initial values).
   - PGD-type method and MU is able to terminate after a few steps. However, the solution is not satisfactory in terms of the relative errors, indicating the algorithm falls into a local minima.
   - On the contrary, the active set method is able to find better solution, despite more iterations are needed.
2) If we take random matrix as initial vakues, all methods will be able to converge to be close to the global minima. However, the active-set methods still outperforms the PGD-type methods.

In addition, we can discover the following phenonmena:

- PGD_AD outperforms PGD_Direct. Hence, it favors the alternating scheme.

[1] https://sites.google.com/site/jingukim/home#nmfcode

- Geneally, BB is the best project gradient method. While the FNMA-I is very sensitive to the stepsize α.

Note: In this part, proposed method, CG, is not included. While the code is self-contained. I will update the results of CG later.

a) All-ones init: The PGD type and MU methods take the initial value of all-one matrices, and the active set methods take the initial value of random matrix.

Table IV.1
ALL-ONES INIT, PART 1

Datasets	Method	iter	time	rel_cost	rel_pgn
Rand Mat	MU	2	0.0634762	0.095014	1.25e-07
	PGD-AD	3	0.1671331	0.0950154	3.77e-05
	PGD-Direct	111	5.6326892	0.0950165	9.58e-05
	BB	1	0.0733216	0.0950145	6.30e-05
	FNMA-I	1	0.072128	0.0950149	4.71e-17
	ANLS-AS-GROUP	209	6.6257293	0.0007823	9.97e-05
	ANLS-AS-UPDATE	180	40.1296417	0.0010351	9.98e-05
	ANLS-BPP	334	11.5577765	0.0006917	9.93e-05
Yale 32 x 32	MU	3	0.0244412	0.4003346	4.51e-06
	PGD-AD	6	0.0602825	0.4003346	8.78e-05
	PGD-Direct	340	2.6615754	0.4003347	9.95e-05
	BB	5	0.045666	0.4003346	5.31e-05
	FNMA-I	1	0.0144713	0.4004964	1.67e-15
	ANLS-AS-GROUP	220	5.1886074	0.2263192	9.91e-05
	ANLS-AS-UPDATE	234	18.1851141	0.2263076	9.91e-05
	ANLS-BPP	129	2.3321463	0.2262807	9.96e-05
Yale 64 x 64	MU	2	0.0448142	0.3836777	9.39e-05
	PGD-AD	5	0.1273182	0.3836776	5.63e-05
	PGD-Direct	3730	78.0896153	0.3836777	9.91e-05
	BB	4	0.1110037	0.3836776	1.21e-05
	FNMA-I	1	0.0517519	0.3838597	1.73e-15
	ANLS-AS-GROUP	82	3.2363882	0.2025569	9.89e-05
	ANLS-AS-UPDATE	93	21.3143733	0.2030481	9.98e-05
	ANLS-BPP	95	3.7758855	0.2027377	9.99e-05

Table IV.2
ALL-ONES INIT, PART 2

Datasets	Method	iter	time	rel_cost	rel_pgn
OCR 32 x 32	MU	2	0.0179377	0.2128464	8.61e-06
	PGD-AD	6	0.1269463	0.2128464	1.36e-05
	PGD-Direct	69	1.2389836	0.2128465	9.49e-05
	BB	2	0.0384127	0.2128464	4.74e-05
	FNMA-I	1	0.0256251	0.2128731	2.23e-15
	ANLS-AS-GROUP	119	4.3202804	0.1123091	9.94e-05
	ANLS-AS-UPDATE	116	13.1514948	0.1122206	9.87e-05
	ANLS-BPP	155	6.0171304	0.1122419	9.96e-05
OCR 32 x 32	MU	2	0.0679501	0.2138612	5.35e-06
	PGD-AD	6	0.3982755	0.2138612	7.28e-06
	PGD-Direct	743	40.701225	0.2138612	9.83e-05
	BB	4	0.6401783	0.2138612	1.62e-05
	FNMA-I	1	0.129258	0.2138894	2.41e-15
	ANLS-AS-GROUP	65	8.1742325	0.1136634	9.84e-05
	ANLS-AS-UPDATE	89	33.4902851	0.1139554	9.84e-05
	ANLS-BPP	72	6.84843	0.1136423	1.00e-04
BBC sports	MU	3	0.1749529	0.9377335	4.00e-05
	PGD-AD	3	0.2106645	0.9538859	9.89e-05
	PGD-Direct	0	0.0644785	0.9897066	2.17e-06
	BB	1	0.068774	0.9537122	6.08e-05
	FNMA-I	1	0.132299	0.9408308	6.88e-19
	ANLS-AS-GROUP	53	8.858645	0.7764368	9.08e-05
	ANLS-AS-UPDATE	55	35.1259396	0.7759689	9.70e-05
	ANLS-BPP	35	4.9804819	0.7761014	9.21e-05
Text Mining	MU	7	0.2343001	0.5824757	2.80e-05
	PGD-AD	8	0.3874955	0.5826104	9.77e-05
	PGD-Direct	5000	413.1357975	0.5121442	0.057971158
	BB	5	0.4970949	0.5826104	6.65e-05
	FNMA-I	1	0.1569973	0.7646227	3.58e-18
	ANLS-AS-GROUP	89	7.123137	0.1938839	9.91e-05
	ANLS-AS-UPDATE	122	47.2511765	0.1943139	9.89e-05
	ANLS-BPP	106	6.7672186	0.1943144	9.92e-05

b) All-random init: All the method take the initial value of random matrix.

Table IV.3
ALL-RANDOM INIT, PART 1

Datasets	Method	iter	time	rel_cost	rel_pgn
Rand Mat	MU	5000	1.7994151	0.00306886	0.001436481
	PGD-AD	5000	3.3497651	0.00310003	0.001464601
	PGD-Direct	5000	2.2137068	0.00331001	0.001497765
	BB	236	0.4001744	0.00186104	9.92e-05
	FNMA-I	2107	4.1017117	0.00312362	1.00e-04
	ANLS-AS-GROUP	418	0.9321975	0.00182876	9.95e-05
	ANLS-AS-UPDATE	404	2.4787843	0.00144654	9.99e-05
	ANLS-BPP	247	0.4886891	0.00253442	9.93e-05
Yale 32 x 32	MU	5000	1.7994151	0.00306886	0.001436481
	PGD-AD	5000	3.3497651	0.00310003	0.001464601
	PGD-Direct	5000	2.2137068	0.00331001	0.001497765
	BB	236	0.4001744	0.00186104	9.92e-05
	FNMA-I	5000	57.0821108	0.22646493	0.000123439
	ANLS-AS-GROUP	173	3.195197	0.22613218	9.95e-05
	ANLS-AS-UPDATE	188	9.3263954	0.22614243	9.86e-05
	ANLS-BPP	154	2.3679411	0.2265607	9.86e-05
Yale 64 x 64	MU	5000	54.3470377	0.20286028	0.435776943
	PGD-AD	5000	111.1331754	0.20257292	0.085781597
	PGD-Direct	5000	91.9654021	0.26682018	0.727151753
	BB	478	20.3077045	0.20254017	9.96e-05
	FNMA-I	5000	190.5334911	0.203118	0.001086616
	ANLS-AS-GROUP	92	3.305792	0.20301822	9.71e-05
	ANLS-AS-UPDATE	97	18.6672476	0.20293399	9.88e-05
	ANLS-BPP	112	3.9598909	0.20239664	9.77e-05

Table IV.4
ALL-RANDOM INIT, PART 2

Datasets	Method	iter	time	rel_cost	rel_pgn
OCR 32 x 32	MU	5000	42.2873148	0.11322416	0.068349949
	PGD-AD	5000	63.4033838	0.11196467	0.034785003
	PGD-Direct	5000	75.8243075	0.14141454	0.123123226
	BB	469	17.0553477	0.11211716	9.95e-05
	FNMA-I	5000	143.9016236	0.1123497	0.000223764
	ANLS-AS-GROUP	120	3.9733914	0.11234351	9.91e-05
	ANLS-AS-UPDATE	110	10.9767672	0.11227023	9.89e-05
	ANLS-BPP	141	4.4199057	0.11245039	9.93e-05
OCR 32 x 32	MU	5000	143.4988413	0.11417625	0.078128545
	PGD-AD	5000	238.2871503	0.1132826	0.019821524
	PGD-Direct	5000	279.6411713	0.14063169	0.131732057
	BB	395	39.9729803	0.11296143	9.93e-05
	FNMA-I	5000	490.3753745	0.1131756	0.000137304
	ANLS-AS-GROUP	76	6.60048	0.11398218	9.96e-05
	ANLS-AS-UPDATE	56	18.8935958	0.11411991	9.93e-05
	ANLS-BPP	67	5.3393015	0.11395586	9.95e-05
BBC sports	MU	5000	218.01866	0.77587115	0.005509941
	PGD-AD	5000	247.5215913	0.77666761	0.001226171
	PGD-Direct	0	0.2240347	0.98855107	4.25e-05
	BB	11	0.9722354	0.78327213	7.56e-05
	FNMA-I	68	6.9055897	0.78639675	9.90e-05
	ANLS-AS-GROUP	59	8.7277818	0.77586147	9.40e-05
	ANLS-AS-UPDATE	48	29.0964178	0.77610309	9.28e-05
	ANLS-BPP	53	6.5828945	0.77610953	9.79e-05
Text Mining	MU	5000	202.4232239	0.1938848	0.01067249
	PGD-AD	5000	263.7314195	0.1938804	0.065931485
	PGD-Direct	5000	386.2081692	0.1962875	0.019447896
	BB	116	10.1005863	0.1938949	1.16e-05
	FNMA-I	134	9.6698864	0.1939238	9.94e-05
	ANLS-AS-GROUP	152	8.8543381	0.1943118	9.98e-05
	ANLS-AS-UPDATE	187	70.0897224	0.1943113	9.24e-05
	ANLS-BPP	81	4.7091249	0.1943113	9.96e-05

REFERENCES

[Han et al., 2010] Han, L., Neumann, M., and Prasad, U. (2010). Alternating projected barzilai-borwein methods for nonnegative matrix factorization. *Electronic transactions on numerical analysis ETNA*, 36:54–82.

[Kim and Park, 2011] Kim, J. and Park, H. (2011). Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM J. Sci. Comput.*, 33(6):3261–3281.

[Lee and Seung, 2001] Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press.

[Lin, 2007] Lin, C. (2007). Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779.

[Liu and Storey, 1991] Liu, Y. and Storey, C. (1991). Efficient generalized conjugate gradient algorithms, part 1: Theory. *Journal of Optimization Theory and Applications*, 69(1):129–137.

[Raydan, 1997] Raydan, M. (1997). The barzilai and borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7.

[Sra and Dhillon, 2007] Sra, S. and Dhillon, I. (2007). Fast newton-type methods for the least squares nonnegative matrix approximation problem.