Nonnegative Matrix Factorization Using Projected Gradient Algorithms with Sparseness Constraints

Nasser Mohammadiha Arne Leijon
Dept. of Electrical Engineering, KTH
Stockholm, Sweden
{nasser.mohammadiha, arne.leijon}@ee.kth.se

Abstract—*Recently projected gradient (PG) approaches have found many applications in solving the minimization problems underlying nonnegative matrix factorization (NMF). NMF is a linear representation of data that could lead to sparse result of natural images. To improve the parts-based representation of data some sparseness constraints have been proposed. In this paper the efficiency and execution time of five different PG algorithms and the basic multiplicative algorithm for NMF are compared. The factorization is done for an existing and proposed sparse NMF and the results are compared for all these PG methods. To compare the algorithms the resulted factorizations are used for a hand-written digit classifier.*

Keywords—*non-negative matrix factorization, projected gradient algorithms, sparseness*

I. Introduction

Nonnegative matrix factorization (NMF) finds nonnegative factors A and X such that $Y \cong AX$, given the observation matrix Y where $A \in R^{I\times J}$, $X \in R^{J\times T}$ and $Y \in R^{I\times T}$. Depending on an application, the estimated factors may have different interpretations. For example NMF can be used as a method for decomposing an image into parts-based representations in which columns of A are basis vectors (images) and columns of X are mixing coefficients. NMF has found a variety of applications in different fields including image processing and pattern classification [1], [2], [3], [4], [5].

To do the factorization we have to define a cost function and minimize it. There are many possibilities for defining the cost function $D(Y\|AX)$, and many procedures for performing its alternating minimizations. The most widely used cost function is the squared Frobenius norm for which the cost function becomes

$$D(Y\|AX) = \frac{1}{2}\|Y-AX\|_F^2 = \frac{1}{2}\sum_{i=1}^{I}\sum_{t=1}^{T}|Y_{it}-[AX]_{it}|^2 \quad (1)$$

for nonnegative A and X matrices.
Indeed this problem is non-convex and may have several local minima but by fixing one of the matrices the problem will become convex. The basic multiplicative algorithm for NMF [6] uses the update equations (2).

$$\begin{aligned} A_{ij} &\leftarrow A_{ij}\frac{[YX^T]_{ij}}{[AXX^T]_{ij}}, \qquad \forall\, i,j \\ X_{jt} &\leftarrow X_{jt}\frac{[A^TY]_{jt}}{[A^TAX]_{jt}}, \qquad \forall\, j,t \end{aligned} \quad (2)$$

This algorithm is slowly convergent for large scale problems and there is a need to search for more fast and efficient algorithms. One of the popular approaches is applying projected gradient (PG) algorithms which have additive updates. The multiplicative and PG algorithms are special cases of a general framework called "Alternating Non-Negative Least Squares (ANNLS)" which is shown below:

```
1−$Initialize \quad X^1 \geq 0 \quad and \quad A^1 \geq 0$
2−$For \quad k = 1, 2, ...$% outer loop
    $X^{k+1} = argmin_{X\geq 0} D(Y\|A^kX^k)$   % inner loop   (3)
    $A^{k+1} = argmin_{A\geq 0} D(Y\|A^kX^{k+1})$ % inner loop   (4)
```

We refer to (3) and (4) as the subproblems of the ANNLS algorithm.
One of the most useful properties of NMF is that it usually produces a sparse representation of the data. A constraint based on the relation between L_1 and L_2 norm is proposed in [7]. In [8] a faster algorithm is introduced for the same approach and also by direct controlling of the number of nonzero elements (L_0) a new constraint is given.

In this paper we will investigate mainly two concepts: firstly the difference of PG algorithms for NMF in terms of efficiency and execution time is investigated. Secondly sparse NMF and the effect of sparseness in the efficiency and execution time of factorization is studied. By giving some comments on the sparsity of NMF a new sparse NMF is inferred. The efficiency of the new sparse NMF is compared with the old ones. To compare the efficiency of different algorithms we have used the basis matrix, A, for a classification problem and we have used hand-written digit classifier for this purpose. The comparisons are based on the error rate. The simulations show that some of the algorithms are more efficient, faster and robust against parameter changing such as number of iterations and model complexity (number of basis vectors) than the others.

II. Projected Gradient Algorithms

In the following we need some new notations that are defined here. y_t is an observed vector or the t^{th} column of the observation matrix Y, a_j is the j^{th} column of A and x_t is the t^{th} column of X. y'_i is the i^{th} column of Y^T, a'_i is the i^{th}

column of A^T and x'_j is the j^{th} column of X^T so we have

$$A = [a_1, ...a_J] \in R^{I\times J},\ A^T = [a'_1, ...a'_I] \in R^{J\times I}$$
$$X = [x_1, ...x_T] \in R^{J\times T}, X^T = [x'_1, ...x'_J] \in R^{T\times J}$$
$$Y = [y_1, ...y_T] \in R^{I\times T},\ Y^T = [y'_1, ...y'_I] \in R^{T\times I}$$

Two subproblems associated with the ANNLS algorithm used in NMF can be written as equations (5) and (6).

$$min_{x_t\geq 0}\ \{D(y_t\|Ax_t) = \frac{1}{2}\|y_t - Ax_t\|_2^2\},\ t = 1, ..., T \quad (5)$$
$$min_{a'_i\geq 0}\ \{D(y'_i\|X^T a'_i) = \frac{1}{2}\|y'_i - X^T a'_i\|_2^2\},\ i = 1, ..., I \quad (6)$$

In the matrix form we have (7) and (8).

$$min_{X\geq 0}\ \{D(Y\|AX) = \frac{1}{2}\|Y - AX\|_F^2\} \quad (7)$$
$$min_{A\geq 0}\ \{D(Y^T\|X^T A^T) = \frac{1}{2}\|Y^T - X^T A^T\|_F^2\} \quad (8)$$

Later in this section we will need the gradient vectors and gradient matrices of the cost function $g_X(x_t)$, $G_X(X)$, $g_A(a'_i)$, $G_A(A^T)$ which are defined as

$$g_X(x_t) = \nabla_{x_t} D(y_t\|Ax_t) = A^T(Ax_t - y_t)$$
$$G_X(X) = \nabla_X D(Y\|AX) = A^T(AX - Y)$$

$$g_A(a'_i) = \nabla_{a'_i} D(y'_i\|Aa'_i) = X(X^T a'_i - y'_i)$$
$$G_A(A^T) = \nabla_{A^T} D(Y^T\|X^T A^T) = X(X^T A^T - Y^T)$$

Projected gradient methods can be generally expressed by iterative additive update rules (9)

$$\begin{aligned} X^{k+1} &= P_\Omega[X^k - \eta_X^k \rho_X^k] \\ A^{k+1} &= P_\Omega[A^k - \eta_A^k \rho_A^k] \end{aligned} \quad (9)$$

where $P_\Omega(z)$ is a projection of z onto a convex feasible set and can be done by replacing all the negative entries in z by zero values. ρ is the descent direction and η is the learning rate. Depending on the selection of ρ and η different PG algorithms can be derived.
In the following we will discuss the algorithms to update matrix X and for updating matrix A the transposed system ($Y^T = X^T A^T$) can be solved in a similar way. We also ignore the subscripts for ρ and η, in order to keep the notation uncluttered.

A. Oblique Projected Landweber Method (OPL)

It is well known that ordinary gradient search methods without constraint converge for $\eta \in (0, \eta_{max})$ and

$$\eta_{max} = \frac{2}{\lambda_{max}(A^T A)} \quad (10)$$

In (10) $\lambda_{max}(A^T A)$ is the maximum eigenvalue of $A^T A$. Using the nonnegative property of matrix A, in the OPL method ([9], [10]) learning rate is calculated as: $\eta_j = \frac{2}{(A^T A 1_J)_j}$, where $1_J = [1, ..., 1]^T \in R^J$. Thus the update rule can be expressed by (11) .

$$X^{k+1} = P_\Omega[X^k - diag(\eta_j) G_X^k] \quad (11)$$

B. Lin-Projected Gradient (LPG)

Like OPL, this approach takes gradient as the descent direction. Here the learning rate $\eta^k = \beta^{t_k}$ and t_k is the first non-negative integer t that satisfies

$$D(Y\|AX^{k+1}) - D(Y\|AX^k) \leq \sigma \nabla_X D(Y\|AX^k)^T (X^{k+1} - X^k) \quad (12)$$

This approach is based on the Armijo rule [11] and equation (12) ensures the sufficient decrease of the function value per iteration [12]. The parameters $\beta \in (0, 1)$ and $\sigma \in (0, 1)$ decide about the convergence speed. A common choice is: $\beta = 0.1$ and $\sigma = 0.01$. Lin proposed an improved algorithm for this approach that is faster [13] and we used this algorithm in our comparisons.

C. Barzilai-Borwein Gradient Projection (GPSR-BB)

The Barzilai-Borwein gradient projection method [14] is based on the Quasi-Newton approach where $\rho^k = H_k^{-1} \nabla_X D(Y\|A^k X^k)$ and H_k is the hessian matrix. Barzilai and Borwein proposed an approximation for calculating hessian: $H_k = n_k I$ where n_k is chosen so that this approximation has similar behavior to the true hessian over the most recent step:

$$\nabla_X D(Y\|A^k X^k) - \nabla_X D(Y\|A^k X^{k-1}) \approx n_k(X^k - X^{k-1})$$

and n_k is chosen to satisfy this relationship in the least-squares sense. Since $D(Y\|A^k X^k)$ is a quadratic function, the line search parameter (or learning rate) can be derived in a closed form as (13) and it is limited to be between 0 and 1.

$$\begin{aligned} \eta^k &= \frac{diag\{(\triangle^k)^T \nabla_X D(Y\|AX)\}}{diag\{(\triangle^k)^T A^T A \triangle^k\}} \\ \triangle^k &= P_\Omega[X^k - n_k^{-1} \nabla_X D(Y\|A^k X^k)] - X^k \end{aligned} \quad (13)$$

$\triangle^k$ is the taken step size in the iteration k if we had unit learning rate.

D. Projected Sequential Subspace Optimization (PSOP)

This approach performs minimization over the subspace spanned by several directions to update the columns of X separately [15]. In contrast to the previous algorithms, instead of one direction PSOP uses the current gradient $g^k = g(x_t^k)$, the direction $d_1^k = x_t^k - x_t^0$, the gradient from the P previous iterations $G^P = [g^{k-P}, g^{k-P+1}, ..., g^{k-1}]$ and the linear combination of the P previous gradients $d_2^k = \sum_{n=k-P+1}^{k} w_n g^{n-1}$ with the coefficients w_n defined as

$$w_n = \begin{cases} 1 & if n = 1 \\ \frac{1}{2} + \sqrt{\frac{1}{4} + w_{n-1}^2} & if n > 1 \end{cases}$$

The learning rate can be derived in a closed form again as

$$\eta^k = -((D^k)^T A^T A D^k + \lambda T)^{-1} (D^k)^T \nabla_{x_t} D(y_t \| A x_t) \quad (14)$$

In (14) $D^k \in R^{J \times (P+3)}$ is the matrix containing all the mentioned directions. We set $P = 3$ in our implementation. λ is the regularization parameter which can be set to a very small constant to avoid numerical problems.

E. Sequential Coordinate-Wise Algorithm (SCWA)

The NMF problem (5) can be expressed in terms of a quadratic problem as (15)

$$min_{x_t} \quad \Psi(x_t) \qquad t = 1, \dots, T \quad (15)$$

where

$$\psi(x_t) = \frac{1}{2} x_t^T H x_t + c_t^T x_t, \quad H = A^T A, \quad c_t = -A^T y_t$$

The sequential coordinate-wise algorithm solves the QP problem given by (15) and updates only one single variable X_{jt} in one iteration step. (15) can be easily expanded to get the following update rules [16]:

$$\begin{aligned} X_{jt}^{k+1} &= max(0, X_{jt}^k - \frac{\lambda_j^k}{(A^T A)_{jj}}) \\ X_{pt}^{k+1} &= X_{pt}^k \quad \forall\, p \in \nu \setminus \{j\} \\ \lambda_t^{k+1} &= \lambda_t^k + (X_{jt}^{k+1} - X_{jt}^k) h_j \end{aligned}$$

hear h_j is the j^{th} column of H, $\lambda_t \in R^J$ is a vector of Lagrange multipliers and $\nu = \{1, ..., J\}$.

III. Adding Sparseness Constraint to NMF

In this paper we used a sparseness measure based on the relationship between L1 and L2 norm:

$$sparseness(x) = \frac{\sqrt{n} - (\sum |x_i|)/\sqrt{\sum x_i^2}}{\sqrt{n} - 1} \quad (16)$$

where n is the dimensionality of vector x. This function gives a value between 0 and 1 depending on the sparseness of x.
Now we have to decide about the sparse factorization: which one of the matrices in the NMF are sparse, A or X? The choice depends on the application mainly. If basis vectors, columns of A, are sparse they affect a small part of each observation. If columns of X are sparse each observation is approximated by a linear combination of a limited number of basis vectors. If rows of X are sparse, then each basis vector is used to approximate a limited number of training data or limited number of training data are used to infer each basis vector. To learn useful features from a database of images usually people constrain the NMF to have columns of A and rows of X sparse ([7], [17]). Having sparse basis vectors makes good sense, but it is better to use all of the training data to derive each basis vector, so it is more useful to let the optimization algorithm decide about the sparseness of rows of X. If we are using a complex model with a large number of basis vectors then it is better to have an X matrix which has sparse columns. In this way a limited number of basis vectors are present for each observation. But if the model is simple then the imposed sparseness on the columns of X does not improve the efficiency of the factorization. To impose the constraints to the PG algorithms we use the alternating nonnegative least squares (ANNLS) algorithm again to have the following general algorithm:

```
1-$Initialize \quad X^1 \geq 0 \quad and \quad A^1 \geq 0$
2-$For \quad k = 1, 2, ...$% outer loop
    $X^{k+1} = argmin_{X \geq 0} D(Y \| A^k X^k)$   %  inner loop
    Apply sparsness projection to $X^{k+1}$
    $A^{k+1} = argmin_{A \geq 0} D(Y \| A^k X^{k+1})$   %  inner loop
    Apply sparsness projection to $A^{k+1}$
```

The projection is applied to columns or rows of input to force them to have the required degree of sparseness by using (16).

IV. Experiments

To compare the PG algorithms with/without sparseness constraint we use them in a classification problem to classify hand-written digits: We have T images of each digit. Each image is stored in a column vector and has I pixels. By combining the training images for all the digits we have observation matrix $Y \in R^{I \times 10T}$. Let us have J basis vectors for each digit so $A \in R^{I \times 10J}$. After performing NMF, feature vectors are obtained by multiplying the pseudo-inverse of A to Y:

$$\Phi = (A^T A)^{-1} A^T Y, \qquad \Phi \in R^{10J \times 10T} \quad (17)$$

Now for any given new test image y we calculate the associated feature vector ϕ by using y instead of Y in (17). The Euclidean distance between ϕ and all the feature vectors in the database, columns of Φ, are calculated. The associated digit to the feature vector that gives the minimum distance is chosen as the result of classifier. Having τ test images of each digit the error rate is calculated as (18)

$$\text{error rate} = 1 - \frac{\text{number of correct recognitions}}{10 \times \tau} \quad (18)$$

Algorithms are tested with the Monte Carlo (MC) analysis, running each one 20 times. Identical initializations are applied to all the PG algorithms and every iteration is initialized randomly with random selection of training and testing database from the standard hand-written data base US Postal Service (USPS). Each image has $I = 256$ pixels and we have chosen

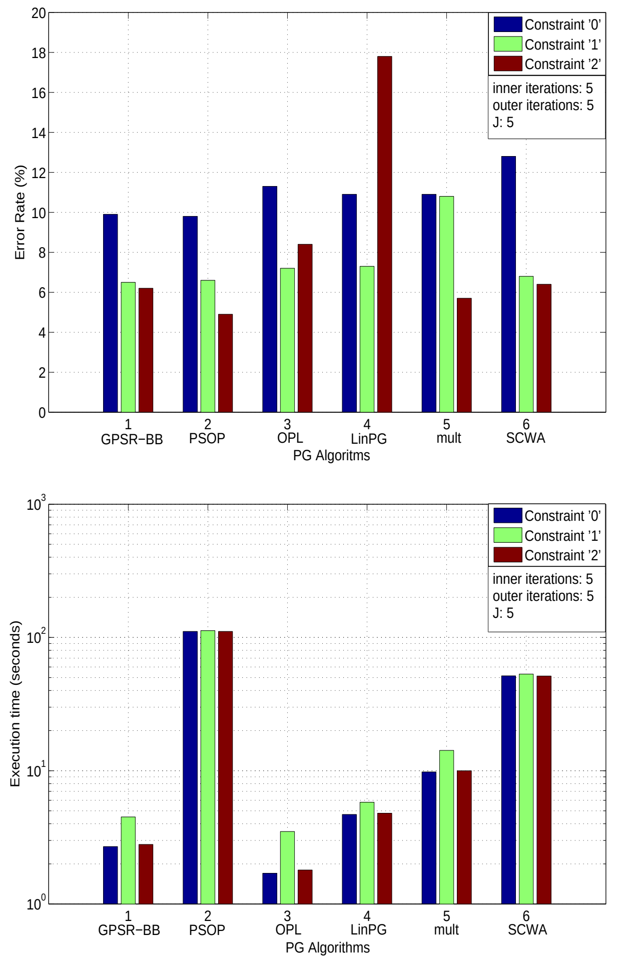

Fig. 1. *Comparison of different PG algorithms for a simple model with different sparseness constraints. constraint '0': no sparse constraint; constraint '1': spareness is applied to the columns of A and rows of X; constraint '2': sparseness is applied to the columns of A only. 'J' is the number of basis vectors per digit. 'inner iterations' is the number of iterations for inner loop and 'outer iterations' is the number of iterations for outer loop; see Alternating Non-Negative Least Squares (ANNLS) algorithm.*

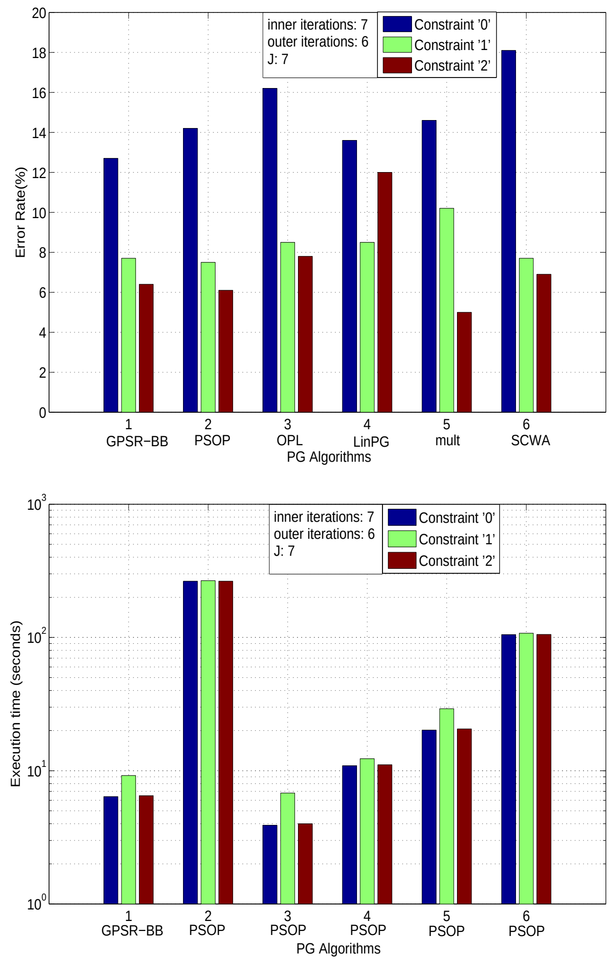

Fig. 2. *Comparison of different PG algorithms for a complex model with different sparseness constraints. constraint '0': no sparse constraint; constraint '1': spareness is applied to the columns of A and rows of X; constraint '2': sparseness is applied to the columns of A only. 'J' is the number of basis vectors per digit. 'inner iterations' is the number of iterations for inner loop and 'outer iterations' is the number of iterations for outer loop; see Alternating Non-Negative Least Squares (ANNLS) algorithm.*

$T = 1000$ and $\tau = 100$.

Fig. 1 and Fig. 2 show the error rate and execution time for different algorithms. Each algorithm is run for three different spareness constraints.

- Constraint '0': no sparse constraint.
- Constraint '1': spareness is applied to the columns of A and rows of X.
- Constraint '2': sparseness is applied to the columns of A only.

A simple model ($J = 5$) is shown in Fig. 1 and a more complex model ($J = 7$) is shown in Fig. 2. The multiplicative algorithm is also shown in the figures for comparison. For this algorithm equation (2) was repeated for (inner iterations$\times$ outer iterations) times.

Some important points can be inferred by looking at these figures:

- The least error rate is obtained using the PSOP algorithm but on the hand side it is the slowest algorithm. GPSR-BB gives a small error rate as well as fast factorization. The multiplicative algorithm gives a good decomposition but it is slower than the PG algorithms with the same efficiency (like GPSR-BB).
- Almost for all of the algorithms there is a large decrease in the error rate by imposing a spareness constraint to the NMF derivation. For example, for the PSOP algorithm in the simple model (Fig. 1) there is a decrease by 3.2 percentage points (pp) in the error rate by using constraint '1' and 4.9 pp decrease in the error rate by

Table 1. Comparison of different PG algorithms with different parameters. Suffix '0' is used to show constraint '0' (no sparse constraint) and suffix '2' is used to show constraint '2' (sparseness is applied to the columns of A only). 'J' is the number of basis vectors per digit. 'i' is the number of iterations for inner loop and 'o' is the number of iterations for outer loop; see Alternating Non-Negative Least Squares (ANNLS) algorithm.

	i:4 o:6 J:5	i:5 o:7 J:5	i:6 o:8 J:5	i:8 o:10 J:5	i:4 o:6 J:8	i:6 o:8 J:8	i:8 o:10 J:8	i:10 o:12 J:8
GPSR-BB0	9.9	9.0	8.3	7.4	18.0	13.2	10.9	9.2
GPSR-BB2	6.3	5.6	5.2	4.9	9.2	7.4	6.7	6.6
PSOP0	9.1	9.4	9.5	9.3	17.0	16.2	15.6	14.2
PSOP2	5.0	4.7	4.8	4.8	7.0	7.0	7.0	6.5
OPL0	11.8	10.0	9.8	8.6	24.8	18.6	15.7	12.6
OPL2	5.7	6.9	5.3	5.0	8.2	7.5	7.6	7.1
LinPG0	12.4	9.4	8.7	7.3	39.6	21.3	16.2	14.7
LinPG2	20.8	13.4	13.5	5.4	26.7	17.3	7.5	7.2
mult0	11.0	10.4	9.3	7.7	19.8	16.4	13.3	11.1
mult2	5.8	5.0	4.6	4.4	5.2	5.1	5.4	5.7
SCWA0	13.7	13.5	12.9	12.3	22.4	23.0	22.8	22.6
SCWA2	6.3	6.1	5.9	5.7	7.5	7.4	7.7	7.3

using constraint '2'. It is worth to mention that as we claimed in the previous section, imposing constraint '2' results to a larger reduction in the error rate than imposing constraint '1', since in the case of constraint '1' a limited number of training images are used to infer each basis vector.

- Imposing a sparseness constraint needs some extra time which is about 1.8 s for constraint '1' and 0.1 s for constraint '2' (for the simple model). This is a relatively small time for slow algorithms but for fast algorithms it becomes relatively large. For example for the OPL algorithm applying constraint '1' almost doubles the execution time while the constraint '2' increases the execution time by 5 percent.

To compare the robustness of algorithms the error rates for different sets of parameters are summarized in Table 1. For each algorithm the result for the basic NMF and NMF having sparse basis vectors are shown. A simple model is also compared versus a complex one in this Table. For the simple model we have used 5 basis vectors for each digit and for the complex one 8 basis vectors are used. Again we can infer some important points from this Table:

- The error rate is decreased by using the spareness constraint '2'. For example, for the simple model an average decrease by 4.5 pp in the error rate is obtained by the PSOP algorithm and in some cases the decrease is about 10 pp in the error rate.
- Among the PG algorithms the best and stable factorizations are obtained by using the PSOP and GPSR-BB algorithms.
- Consider the results for each algorithm separately, it can be seen that the sparseness has improved the robustness and the results are less sensitive to parameter changing. In other words the sparseness constraint decreases the chance of getting stuck in a local minimum. For example consider the OPL algorithm in Table 1. The average error rate is 14% for the basic NMF (OPL0) and for the sparse NMF (OPL2) it is 6.7%. The standard deviation is 5.5 for the OPL0 where for the OPL2 it is 1.2. For another example consider the multiplicative algorithm: without constraint the error rate is high and is changing a lot but by imposing the constraint the result becomes much better and stable.
- We can also compare two models using Table 1. As we see the complex model with 120 iterations (column 9 in Table 1) does not give a lower error rate than the simple model with 35 iterations (column 3 in Table 1). The reason is that the complex model takes into account the noise in the data and it is getting closer to the over fitting where it can not represent test data well. So this experience shows the hidden disadvantage of the complex model.

V. Conclusions

Different implemented PG algorithms are used for a hand-written digit recognizer and their performance are compared. The results show that the PSOP and GPSR-BB algorithms are more efficient and stable than the others. Although the PSOP algorithm is the slowest algorithm, the GPSR-BB algorithm is the second fastest one. Basic NMF is compared with sparse NMF and the experience shows that imposing a constraint on the basis vectors (columns of A) and rows of X decreases the error rate. It was also seen that the best performance is obtained by imposing the constraint on the basis vectors (columns of A) only where the error rate decreases up to 10 pp with respect to the basic NMF. Finally, we saw that the complex model does not necessarily improve the performance. Choosing the appropriate number of components is dependent to the application and it is important to work on this to find a model comparison framework for NMF.

Acknowledgments

This work was supported by the EU Initial Training Network AUDIS (grant 2008-214699).

References

[1] V. K. Potluru and V. D. Calhoun, "Group learning using contrast NMF: application to functional and structural MRI of schizophrenia," in *IEEE International Symposium on Circuits and Systems*, 2008, pp. 1336–1339.
[2] M. Rajapakse, J. Tan, and J. Rajapakse, "Color channel encoding with NMF for face recognition," in *IEEE International Conference on Image Processing*, 2004, pp. 2007–2010.
[3] A. B. J. Teoh, H. F. Neo, and D. C. L. Ngo, "Sorted locally confined non-negative matrix factorization in face verification," in *IEEE International Conference on Communications, Circuits and Systems*, 2005, pp. 820–824.
[4] M. W. Spratling, "Learning image components for object recognition," *Journal of Machine Learning Research*, vol. 7, pp. 793–815, 2006.

[5] W. Liu and N. Zheng, “Nonnegative matrix factorization based methods for object recognition,” *Pattern Recognition Letters*, vol. 25, no. 8, pp. 893–897, 2004.

[6] C.-J. Lin, “On the convergence of multiplicative update algorithms for nonnegative matrix factorization,” *IEEE Transactions on NeuralNetworks*, vol. 18, no. 6, pp. 1589–1596, 2007.

[7] P. O. Hoyer, “non-negative matrix factorization with sparseness constraints,” *Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[8] M. Mørup, K. H. Madsen, and L. K. Hansen, “Approximate l_0 constrained non-negative matrix and tensor factorization,” in *IEEE International Symposium on Circuits and Systems*, 2008, pp. 1328–1331.

[9] M. Bertero and P. Boccacci, *Introduction to Inverse Problems in Imaging*. Institute of Physics, Bristol, UK, 1998.

[10] B. Johanssona, T. Elfvingb, V. Kozlovc, Y. Censord, P.-E. Forssén, and G. Granlunda, “The application of an oblique-projected Landweber method to a model of supervised learning,” *Mathematical and Computer Modelling*, vol. 43, pp. 892–909, 2006.

[11] L. Armijo, “Minimization of functions having Lipschitz continuous first derivatives,” *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[12] D. P. Bertsekas, “On the Goldstein-Levitin-Polyak gradient projection method,” *IEEE Transations on Automatic Control*, vol. 21, no. 2, pp. 174–184, 1976.

[13] C.-J. Lin, “Projected gradient methods for nonnegative matrix factorization,” *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[14] Y. H. Dai and R. Fletcher, “Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming,” *Numerische Mathematik*, vol. 100, no. 1, pp. 21–47, 2005.

[15] G. Narkiss and M. Zibulevsky, *Sequential subspace optimization method for large-scale unconstrained problems*. Tech. Rep. 559, EE, Technion, Israel Institute of Technology, Haifa,Israel, 2005.

[16] V. Franc, V. Hlaváč, and M. Navara, “Sequential coordinatewise algorithm for the nonnegative least squares problem,” in *International Conference on Computer Analysis of Images and Patterns*, 2005, pp. 407–414.

[17] C. Wang, X. Wang, and S. Huang, “Blind watermarking via NMF with sparseness constraint,” in *International Symposium on Communications and Information Technologies*, 2005, pp. 869–872.