Sequential Sparse NMF

Vamsi K. Potluru* Sergey M. Plis Barak A. Pearlmutter
Vince D. Calhoun Thomas P. Hayes

April 1, 2011

Abstract

Nonnegative Matrix Factorization (NMF) is a standard tool for data analysis. An important variant is the Sparse NMF problem. A natural measure of sparsity is the L_0 norm, however its optimization is NP-hard. Here, we consider a sparsity measure linear in the ratio of the L_1 and L_2 norms, and propose an efficient algorithm to handle the norm constraints which arise when optimizing this measure. Although algorithms for solving these are available, they are typically inefficient. We present experimental evidence that our new algorithm performs an order of magnitude faster compared to the previous state-of-the-art.

1 Introduction

Quite a few problems in machine learning and signal processing can be cast as Nonnegative Matrix Factorizations (NMF) which is a special case of low rank approximations. In NMF, given a nonnegative $m \times n$ matrix $\mathbf{X}$, we want to approximate it with a product of two nonnegative matrices $\mathbf{W}, \mathbf{H}$ of sizes $m \times r$ and $r \times n$ respectively:

$$\mathbf{X} \approx \mathbf{WH}. \tag{1}$$

The nonnegative constraint on matrix $\mathbf{H}$ makes the representation a convex combination of features given by $\mathbf{W}$. NMF can result in a parts-based representation and is usually different from other factorization techniques which result in more holistic representations (e.g. Principal Component Analysis (PCA) and Vector Quantization (VQ)). Also, it can be applied to a wide range of nonnegative data for instance image data (Lee and Seung, 1999), biomedical data (Kim and Park, 2007) and text-mining data (Lee and Seung, 1999).

The nonnegative decomposition is in general not unique (Donoho and Stodden, 2004) and can hamper interpretation of the estimated factors. Also, the factors may not be parts-based if, for example, the data resides well inside the positive orthant. To address these issues sparseness constraints have been added to the NMF problem. This leads to the Sparse NMF problem (SNMF).

In section 3, we present our new algorithm to solve the SNMF problem as posed by Hoyer (2004). Implementation details are discussed in section 4. We compare with the algorithm proposed in Hoyer (2004) to demonstrate the effectiveness of our approach and present the results in section 5. Many formulations for SNMF have been proposed and we discuss them in the related work section 6. Section 7 contains conclusions and future work.

2 Preliminaries

In this section, we give an introduction to the NMF and SNMF problems. Also, we discuss some widely used algorithms from the literature to solve them. Note that we use subscripts to denote column indices and superscripts for row indices. When applied to matrices, we get column vectors or row vectors respectively. Simultaneous application gives us the

*Dept. of Computer Science, UNM, New Mexico, USA. Email: ismav@cs.unm.edu

corresponding element of the matrix.

2.1 NMF

Lee and Seung (2001) described simple multiplicative updates for $\mathbf{W}$ and $\mathbf{H}$ corresponding to Frobenius norm of the representation error. The NMF objective is :

$$\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{X}-\mathbf{WH}\|_F^2 \quad \text{s.t.} \mathbf{W} \geq \mathbf{0}, \mathbf{H} \geq \mathbf{0} \tag{2}$$

The multiplicative updates of Lee and Seung (2001) are:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\mathbf{XH}^\top}{\mathbf{WHH}^\top}, \tag{3}$$

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^\top\mathbf{X}}{\mathbf{W}^\top\mathbf{WH}}, \tag{4}$$

where $\odot$ represents element-wise Hadamard product, and division and $\geq$ are also element-wise. It should be noted that the NMF objective to be minimized is convex in either $\mathbf{W}$ or $\mathbf{H}$ but not in both. Lee and Seung (2001) also proved that when the algorithm iterates using the updates (3) and (4), the objective is monotonically non-increasing while satisfying the nonnegative constraints automatically. In practice, a small number is added to the denominator in the updates to avoid division by zero.

Besides multiplicative updates, other methods have been proposed to solve the NMF problem such as projected gradient (Lin, 2007) and block pivoting (Kim and Park, 2008).

2.2 Sparse NMF

Hoyer (2004) extended NMF to include sparsity constraints on one or both of the matrix factors. Sparseness(sp) is defined as follows according to Hoyer:

$$\text{sp}(\boldsymbol{x}) = \frac{\sqrt{s} - \|x\|_1/\|x\|_2}{\sqrt{s}-1} \tag{5}$$

where the vector $\boldsymbol{x}$ is of size s. The sparseness function sp is a surrogate function of the the L_0 norm. Note that L_0 is not a true norm. This can be generalized to matrices of size $m \times n$ by defining it to be sparsity of all its elements when expanded out as a vector of size mn. The range of the function sp is zero to one. If the sparseness function evaluates to zero, it means that all the elements are non-zero. If it is one, then only one element is non-zero. The Sparse NMF problem (Hoyer, 2004) is as follows:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{X}-\mathbf{WH}\|_F^2 \\ \text{s.t.} \mathbf{W} \geq \mathbf{0}, \mathbf{H} \geq \mathbf{0} \\ \text{sp}(\boldsymbol{W}_j) = \alpha, \forall j \in \{1,\cdots,r\} \\ \text{sp}(\boldsymbol{H}^i) = \beta, \forall i \in \{1,\cdots,r\}\end{aligned} \tag{6}$$

where we also allow for either of the sparsity constraints to be inactive but not both.

In addition to the new SNMF formulation, Hoyer (2004) also gave a gradient descent algorithm called Nonnegative Matrix Factorization with Sparseness Constraints(NMFSC) to solve problem (6). Multiplicative updates were used for optimizing the unconstrained matrix factor. Heiler and Schnörr (2006) proposed a new algorithm which also solved this problem by sequential cone programming and utilized general purpose solvers like MOSEK.

3 The Sequential Sparse NMF Algorithm

We present our algorithm which we call **S**equential **S**parse **NMF**(**SSNMF**) as follows:

First, we consider a problem of special form in section 3.1 which turns out to be the core routine of our algorithm and give an efficient, as well as an exact algorithm to solve it.

Second, we give routines to solve subproblems of the SNMF problem based on previous work. This includes the multiplicative updates routine from Lee and Seung (2001).

Third, we give the complete SSNMF (Algorithm 5) to solve SNMF based on the previous two steps.

3.1 Sparse-opt

Let us consider the following problem :

$$\max_{\boldsymbol{y}\geq 0} \boldsymbol{b}^\top \boldsymbol{y} \text{ s.t. } \|\boldsymbol{y}\|_1 = k, \|\boldsymbol{y}\|_2 = 1 \tag{7}$$

where the length of vector $\boldsymbol{b}$ is m.

Consider the following permuted problem :

$$\max_{\boldsymbol{x}\geq 0} \boldsymbol{a}^\top \boldsymbol{x} \text{ s.t. } \|\boldsymbol{x}\|_1 = k, \|\boldsymbol{x}\|_2 = 1 \tag{8}$$

where $\boldsymbol{a} = \text{sort}(\boldsymbol{b})$. By the structure of the permuted problem, it is clear that there is a transition point $p = p^*$ such that all the elements of vector $\boldsymbol{x}$ to the right of index p are zero and the rest positive. Also, we can get the solution $\boldsymbol{y}$ for problem (7) from solution $\boldsymbol{x}$ of the permuted problem (8) by a permutation.

The Lagrangian of the permuted problem is :

$$\max_{\boldsymbol{x},\mu,\lambda} \boldsymbol{a}^\top \boldsymbol{x} + \mu\left(\sum_{i=1}^{m} x_i - k\right) + \frac{\lambda}{2}\left(\sum_{i=1}^{m} x_i^2 - 1\right)$$

Setting the derivatives of the Lagrangian to zero, we get:

$$\begin{aligned} a_i + \mu + \lambda x_i = 0, \forall i \in \{1, 2, \cdots, p\} \\ \sum_{i}^{m} x_i = k \\ \sum_{i}^{m} x_i^2 = 1 \\ x_i = 0, \forall i \in \{p+1, \cdots, m\} \end{aligned}$$

By applying the Cauchy-Schwartz inequality on the vector $\boldsymbol{x}$, we have $p \geq k^2$. We propose a new algorithm called Sparse-opt to solve this problem(Algorithm 1). Hoyer (2004) also considered this problem and gave a heuristic to solve it which we will henceforth refer to as the Projection-heuristic.

Algorithm 1 Sparse-opt($\boldsymbol{b}, k$)

1: Input: Vector $\boldsymbol{b}$ and sparsity k.
2: Set $\boldsymbol{a} = \text{sort}(\boldsymbol{b})$ and get the mapping π such that $a_i = b_{\pi(i)}$ and $a_j > a_{j+1}$ for all valid i, j.
3: Compute values of $\mu(p), \lambda(p)$ and $obj(p))$ for p from $\lceil k^2 \rceil$ to m as follows:

$$\lambda(p) = -\sqrt{\frac{p\sum_i a_i^2 - \left(\sum_i a_i\right)^2}{(p-k^2)}}$$
$$\mu(p) = -\frac{\sum_i a_i}{p} - \frac{k}{p}\lambda(p)$$
$$obj(p) = \frac{-\lambda(p+k) + \sum_i a_i}{p}$$

4: Find the p for which the value of obj which is the largest and call it p^*.
5: Set $x_i = -\frac{a_i + \mu(p^*)}{\lambda(p^*)}, \forall i \in \{1, \cdots, p^*\}$ and to zero otherwise.
6: Generate solution such that $y_{\pi(i)} = x_i$.
7: Output: Vector $\boldsymbol{y}$

3.2 SSNMF

The SNMF problem (6) can also be rewritten as the following:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{X} - \mathbf{WDH}\|_F^2 \\ \text{s.t.} \mathbf{W} \geq \mathbf{0}, \mathbf{H} \geq \mathbf{0}, \mathbf{D} \geq \mathbf{0} \\ \text{sp}(\mathbf{W}_j) = \alpha, \forall j \in \{1, \cdots, r\} \\ \text{sp}(\mathbf{H}^i) = \beta, \forall i \in \{1, \cdots, r\} \end{aligned} \tag{9}$$

where $\mathbf{D}$ is a $r \times r$ diagonal matrix. We use the new formulation to additionally constrain the L_2 norms of the columns of matrix $\mathbf{W}$ if its sparsity value is given. Similarly, we constrain the rows of matrix $\mathbf{H}$ to be 1 if its sparsity value is given. This scaling is absorbed by the matrix diagonal $\mathbf{D}$. Since, scaling is one the properties satisfied by the Hoyer's sparsity criterion (Hurley and Rickard, 2009), we can use our formulation of SNMF and give a solution to Problem (6).

The core multiplicative updates routine for solving the NMF problem proposed by Lee and Seung

(2001) is given in Algorithm 2. This essentially is a multiplicative update algorithm for solving the NNLS problem with multiple right-hand sides. Also, the multiplicative update rule for the diagonal matrix derived by Ding et al. (2006) is given in Algorithm 3. Algorithm 3 and algorithm 2 can be sped up by precomputing the matrix products which are unchanged during the iterations. Consider the objective in problem 9 for a column j of the matrix $\mathbf{W}$ while fixing the rest of the elements of matrices $\mathbf{W}, \mathbf{H}, \mathbf{D}$:

$$f(\mathbf{W}_j) = \frac{1}{2} g \|\mathbf{W}_j\|^2 + \boldsymbol{u}^\top \mathbf{W}_j$$

for fixed quantities $g, \boldsymbol{u}$. The optimization problem is given by:

$$\min_{\mathbf{W}_j \geq \mathbf{0}} f(\mathbf{W}_j) \text{ s.t. } \|\mathbf{W}_j\|_2 = 1, \|\mathbf{W}_j\|_1 = k \qquad (10)$$

This reduces to the problem we considered in section 3.1. We update the columns of the matrix factor $\mathbf{W}$ sequentially as shown in Algorithm 4. Algorithm 4 combined with algorithms 1, 2 and 3 gives us SSNMF to solve SNMF(Algorithm 5). We call it sequential for we update the columns one at a time unlike the batch manner adopted by NMFSC.

Algorithm 2 mult($\mathbf{X}, \mathbf{W}, \mathbf{H}$, *maxiter*)

Input: Matrices $\mathbf{X}, \mathbf{W}, \mathbf{H}$ and positive number *maxiter*
for iter = 1 to *maxiter* **do**
  $\mathbf{H} = \mathbf{H} \odot \frac{\mathbf{W}^\top \mathbf{X}}{\mathbf{W}^\top \mathbf{W} \mathbf{H}}$.
end for
Output: Matrix $\mathbf{H}$.

Algorithm 3 multD($\mathbf{X}, \mathbf{W}, \mathbf{H}, \mathbf{D}$, *maxiter*)

Input: Matrices $\mathbf{X}, \mathbf{W}, \mathbf{H}, \mathbf{D}$ and positive number *maxiter*
for iter = 1 to *maxiter* **do**
  $\mathbf{D} = \mathbf{D} \odot \frac{\mathbf{W}^\top \mathbf{X} \mathbf{H}}{\mathbf{W}^\top \mathbf{W} \mathbf{D} \mathbf{H} \mathbf{H}^\top}$
end for
Output: Matrix $\mathbf{D}$.

Algorithm 4 spar($\mathbf{X}, \mathbf{W}, \mathbf{H}, k$, *maxiter*)

Input: Matrices $\mathbf{X}, \mathbf{W}, \mathbf{H}$, positive numbers *maxiter* and sparsity k.
$\mathbf{C} = -\mathbf{X}\mathbf{H}^\top + \mathbf{W}\mathbf{H}\mathbf{H}^\top$
$\mathbf{G} = \mathbf{H}\mathbf{H}^\top$
for iter=1 to *maxiter* **do**
  for i =1 to r **do**
    $\mathbf{U}_j = \mathbf{C}_j - \mathbf{W}_j G^j_j$
    $\boldsymbol{t}$ = Sparse-opt($-\mathbf{U}_j, k$).
    $\mathbf{C} = \mathbf{C} + (\boldsymbol{t} - \mathbf{W}_i)\, \boldsymbol{G}^i$
    $\mathbf{W}_i = \boldsymbol{t}$.
  end for
end for
Output: Matrix $\mathbf{W}$.

4 Implementation Issues

We describe some of the modifications to the ssnmf algorithm proposed in the previous section.

4.1 Random order of updates

Instead of updating the columns in a sequential order, we propose to use random permutations.

4.2 Incorporating faster solvers

We use multiplicative updates for a fair comparison with NMFSC. However, there have been few improved solvers since them, for example Lin (2007), Kim and Park (2008). We can plug in these solvers pretty easily to solve parts of the SNMF problem when we don't have sparsity constraints on one of the factors.

4.3 Convergence

We measure objective values at the end of an iteration and terminate when relative change in objective value between iterations is less than a pre-defined tolerance value.

Algorithm 5 ssnmf($\mathbf{X}, r, \alpha, \beta, maxiter$)

Input: Matrix $\mathbf{X}$, rank r, number of iterations $maxiter$ and sparsity values k, l.
Initialize $\mathbf{W}$,$\mathbf{D}$ and $\mathbf{H}$ as follows: If sparsity value α is given, generate a positive random vector $\boldsymbol{v}$ of size m and obtain $\boldsymbol{z} = \text{Sparse-opt}(\boldsymbol{v}, k)$ where $k = \sqrt{m} - \sqrt{m-1}\alpha$ (from equation (5)). Use the solution $\boldsymbol{z}$ and its random permutations to initialize matrix $\mathbf{W}$. Similarly, initialize the matrix $\mathbf{H}$. If the sparsity value is not given, just initialize the matrix to uniformly random entries in $[0, 1]$. Set $\mathbf{D}$ to be the identity matrix of size $r \times r$.
repeat
  { Update $\mathbf{W}$}
  if Sparseness constraints apply **then**
    $\mathbf{W} = \text{spar}(\mathbf{X}, \mathbf{W}, \mathbf{DH}, k, maxiter)$
  else
    $\mathbf{Wt} = \text{mult}(\mathbf{X}^\top, \mathbf{H}^\top\mathbf{D}, \mathbf{W}^\top, maxiter)$
    $\mathbf{W} = \mathbf{Wt}^\top$
  end if
  {Update $\mathbf{D}$}
  if Sparsity on both factors **then**
    $\mathbf{D} = \text{multD}(\mathbf{X}, \mathbf{W}, \mathbf{H}, \mathbf{D}, maxiter)$
  end if
  {Update $\mathbf{H}$ }
  if Sparseness constraints apply **then**
    $\mathbf{Ht} = \text{spar}(\mathbf{X}^\top, \mathbf{H}^\top, \mathbf{DW}^\top, l, maxiter)$
    $\mathbf{H} = \mathbf{Ht}^\top$
  else
    $\mathbf{H} = \text{mult}(\mathbf{X}, \mathbf{WD}, \mathbf{H}, maxiter)$
  end if
until convergence
Output: Matrices $\mathbf{W}, \mathbf{D}, \mathbf{H}$.

5 Experiments

In this section, we compare SSNMF with NMFSC. Experiments report reconstructive error instead of objective value for convenience of display. All of our experiments were run on a 2.8Ghz machine with the number of threads set to one. Our algorithm SSNMF was implemented in Matlab similar to NMFSC.

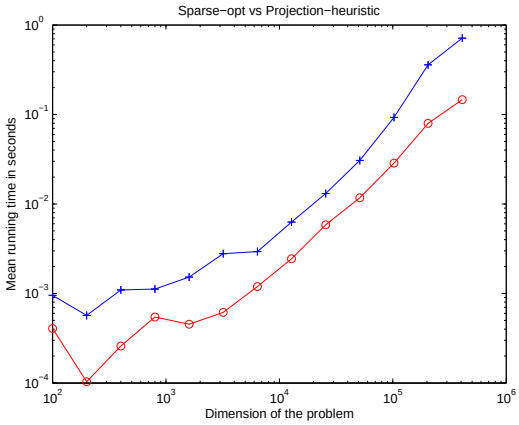

Figure 1: Running times(lower values are better) for Sparse-opt(shown in red circles) vs the Projection-heuristic(shown in blue pluses) for random problems of size $2^i * 50$. The x-axis plots the dimension of the problem i ranges from 100 to $2^{12}*100$ while the y-axis has the running time in seconds.

5.1 Comparing Performances of Core Updates

First, we compare our Sparse-opt algorithm with the Projection-heuristic (Hoyer, 2004). To do this, we generate 40 random problems with sparsity constraint being uniformly random in $[0, 1]$ for a fixed dimension. Also, the input vector is generated by taking uniform random samples from $[0, 1]$. We consider dimensions $2^i * 100$ where i takes integer values from 0 to 12.

The mean values of the running times for Sparse-opt and the Projection-heuristic for each dimension are plotted in Figure 1.

5.2 Datasets

For comparing the performance of SSNMF with NMFSC, we consider the following 2 real-world datasets:

1. **CBCL** face dataset consists of 2429 images of size 19×19 and can be obtained at `http://cbcl.mit.edu/cbcl/`

software-datasets/FaceData2.html.

2. **ORL** dataset consists of 400 images of size 112 × 92 and can be obtained at http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

5.3 Comparing Overall Performances

To ensure fairness, we removed logging information from NMFSC code (Hoyer, 2004) and only computed the objective every 20 iterations. The parameter *maxiter* was set to 10 in SSNMF. Also, we used the objective value in SSNMF as a stopping criterion for NMFSC. We set the number of outer iterations for SSNMF to be 25 as termination condition for all the experiments.

We applied SSNMF and NMFSC on the CBCL face dataset. We use the same preprocessing as done in Hoyer (2004). Rank was set to 49 in both the algorithms. The results are shown in Table 1.

Similarly, we ran SSNMF and NMFSC on the ORL face dataset. The rank was fixed at 25 in both the algorithms. The results are shown in Table 2.

Also, we compare the effect of switching the solver in our SSNMF algorithm. We use the algorithm from Lin (2007) and call the resulting algorithm SSNMF+Lin. Similarly, we replace the Sparse-opt routine in our algorithm by the Projection-heuristic of NMFSC and call the resulting algorithm SSNMF+Proj . The results of running these modified algorithms on the ORL face dataset is shown in Table 3.

6 Related Work

In this paper we derive an algorithm for the SNMF problem as formulated in Hoyer (2004). Other SNMF formulations include Hoyer (2002),L_0-constrained NMF (Mørup, Madsen, and Hansen, 2008), Kim and Park (2007) and Pascual-Montano et al. (2006).

First, we would like to note that the sparsity measure used in this paper has all the desirables properties as discussed extensively in the paper by Hurley and Rickard (2009) except for cloning. This is not an issue for us since the dimensions are fixed in our optimization problem and hence we do not have a problem with the measure not satisfying the cloning property. L_1 measure satisfies only one of the 6 properties given by Hurley and Rickard (2009) namely 'rising tide'. The measure used in Kim and Park (2007) is based on L_1 norm. The properties satisfied by the measure in Pascual-Montano et al. (2006) is unclear because of the implicit nature.

Secondly, the sparsity in both Pascual-Montano et al. (2006) and Kim and Park (2007) use an implicit measure for sparsity. The sparsity measure considered in this paper enables us to handle sparsity explicitly.

Thirdly, Pascual-Montano et al. (2006) claim that the SNMF formulation Hoyer (2004) doesn't capture the variance in the data. We would like to argue that the experiments and the conclusions thereby reached were biased. The sparsity was set on both the matrix factors unlike Pascual-Montano et al. (2006) which has a single sparsity parameter. A fairer comparison would be to set sparsity on one of the factors in Hoyer (2004) and let the other factor compensate to explain the data.

We were unable to compare with the algorithm in Heiler and Schnörr (2006) for the code is not publicly available. However, we believe that the specialized algorithm we developed should be faster than the general purpose solvers used in Heiler and Schnörr (2006).

7 Conclusions and Future Work

We have proposed a new efficient algorithm to solve the SNMF problem. Experiments demonstrate the effectiveness of our approach on real datasets of practical interest. Our algorithm is faster over a range of sparsity values and does especially better when the sparsity is higher. Also, we have proposed an exact algorithm to solve the sparsity constraint unlike the heuristic in Hoyer (2004). Also, experiments show it to be atleast as fast as the heuristic in practice. The speed up is because of the sequential nature of the updates as opposed to the batch approach of Hoyer

Sparsity	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75
SSNMF(seconds)	35.44	35.29	35.26	35.33	35.25	36.21	37.66	38.12	36.33	36.07
NMFSC(seconds)	152.97	195.14	299.74	424.96	411.05	369.65	488.44	652.25	741.14	911.80
Ratio	4.32	5.53	8.50	12.03	11.66	10.21	12.97	17.11	20.40	25.28
Reconstruction Error	5.35	5.32	5.29	5.31	5.33	5.36	5.44	5.51	5.69	6.04

Table 1: Running times(seconds), reconstruction error and corresponding sparsity parameters are shown for the two algorithms for the ORL faces dataset. We used multiplicative updates for updating **H** to ensure fairness in comparison with Hoyer's algorithm. First our algorithm was run and the resulting objective value was used as a stopping criterion for the Hoyer's algorithm to ensure we have the same objective value.

Sparsity	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75
SSNMF(seconds)	6.10	6.06	5.94	5.99	6.03	5.97	6.11	5.99	5.89	5.83
NMFSC(seconds)	45.00	42.66	52.35	46.12	56.37	59.70	71.54	79.69	81.91	129.40
Ratio	7.38	7.04	8.82	7.70	9.35	10.00	11.70	13.30	13.91	22.17
Reconstruction Error	57.91	55.07	52.33	50.32	48.33	46.84	45.12	43.92	42.57	41.52

Table 2: Running times(seconds), reconstruction error and corresponding sparsity parameters are shown for the two algorithms for the CBCL faces dataset. We used multiplicative updates for updating **H** to ensure fairness in comparison with Hoyer's algorithm. First our algorithm was run and the objective value was used as a stopping criterion for the Hoyer's algorithm to ensure we have the same objective value.

(2004).

Our approach can also be potentially applied to other NMF variants which include an additional term in the objective. For example, we could use the framework in this paper to solve the SNMF problem as posed in Hoyer (2002). Also, we can potentially extend this approach to solve sparse versions of semi-NMF and convex NMF as well as the sparse Tensor extensions of NMF .

Acknowledgement

The first author would like to acknowledge the support from NIBIB grants 1 R01 EB 000840 and 1 R01 EB 005846. The second author was supported by NIMH grant 1 R01 MH076282-01. The latter two grants were funded as part of the NSF/NIH Collaborative Research in Computational Neuroscience Program.

Sparsity	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75
SSNMF(seconds)	35.44	35.29	35.26	35.33	35.25	36.21	37.66	38.12	36.33	36.07
SSNMF+Lin(seconds)	15.85	17.41	18.68	19.98	21.33	22.97	24.05	25.41	26.90	28.25
SSNMF+Proj(seconds)	36.70	37.02	37.58	42.68	39.76	38.94	39.50	41.01	42.02	43.45

Table 3: Running times(seconds) and corresponding sparsity parameters are shown for SSNMF and modified versions of it on the ORL faces dataset. Reconstruction error is ensured to be the same or better for the modified algorithms.

References

Ding, C.; Li, T.; Peng, W.; and Park, H. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, 126–135. New York, NY, USA: ACM.

Donoho, D., and Stodden, V. 2004. When does non-negative matrix factorization give a correct decomposition into parts? In Thrun, S.; Saul, L.; and Schölkopf, B., eds., *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

Heiler, M., and Schnörr, C. 2006. Learning sparse representations by non-negative matrix factorization and sequential cone programming. *Journal of Machine Learning Research* 7:2006.

Hoyer, P. O. 2002. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, 557–565.

Hoyer, P. O. 2004. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.* 5:1457–1469.

Hurley, N., and Rickard, S. 2009. Comparing measures of sparsity. *IEEE Trans. Inf. Theor.* 55:4723–4741.

Kim, H., and Park, H. 2007. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics* 23(12):1495–1502.

Kim, J., and Park, H. 2008. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. *Data Mining, IEEE International Conference on* 0:353–362.

Lee, D. D., and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791.

Lee, D. D., and Seung, S. H. 2001. Algorithms for non-negative matrix factorization. In *NIPS*, 556–562.

Lin, C.-J. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Comp.* 19(10):2756–2779.

Mørup, M.; Madsen, K. H.; and Hansen, L. K. 2008. Approximate l0 constrained non-negative matrix and tensor factorization. In *ISCAS*, 1328–1331.

Pascual-Montano, A.; Carazo, J.; Kochi, K.; Lehmann, D.; and Pascual-Marqui, R. 2006. Nonsmooth nonnegative matrix factorization (nsnmf). *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28(3):403 –415.