Using Population Based Algorithms for Initializing Nonnegative Matrix Factorization

Andreas Janecek and Ying Tan

Key Laboratory of Machine Perception (MOE), Peking University
Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, China
andreas.janecek@univie.ac.at, ytan@pku.edu.cn

Abstract. The nonnegative matrix factorization (NMF) is a bound-constrained low-rank approximation technique for nonnegative multivariate data. NMF has been studied extensively over the last years, but an important aspect which only has received little attention so far is a proper initialization of the NMF factors in order to achieve a faster error reduction. Since the NMF objective function is usually non-differentiable, discontinuous, and may possess many local minima, heuristic search algorithms are a promising choice as initialization enhancers for NMF.

In this paper we investigate the application of five population based algorithms (genetic algorithms, particle swarm optimization, fish school search, differential evolution, and fireworks algorithm) as new initialization variants for NMF. Experimental evaluation shows that some of them are well suited as initialization enhancers and can reduce the number of NMF iterations needed to achieve a given accuracy. Moreover, we compare the general applicability of these five optimization algorithms for continuous optimization problems, such as the NMF objective function.

1 Introduction

The nonnegative matrix factorization (NMF, [1]) leads to a low-rank approximation which satisfies nonnegativity constraints. Contrary to other low-rank approximations such as SVD, these constraints may improve the sparseness of the factors and due to the "additive parts-based" representation also improve interpretability [1, 2]. NMF consists of reduced rank *nonnegative* factors $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ with $k \ll min\{m, n\}$ that approximate matrix $A \in \mathbb{R}^{m \times n}$. NMF requires that *all entries* in A, W and H are zero or positive. The nonlinear optimization problem underlying NMF can generally be stated as

$$\min_{W,H} f(W, H) = \min_{W,H} \frac{1}{2} ||A - WH||_F^2. \quad (1)$$

Initialization. Algorithms for computing NMF are iterative and require initialization of the factors W and H. NMF unavoidably converges to local minima, probably different ones for different initialization (cf. [3]). Hence, random initialization makes the experiments unrepeatable since the solution to Equ. (1) is not

Y. Tan et al. (Eds.): ICSI 2011, Part II, LNCS 6729, pp. 307–316, 2011.

unique in this case. A proper non random initialization can lead to faster error reduction and better overall error at convergence. Moreover, it makes the experiments repeatable. Although the benefits of good NMF initialization techniques are well known in the literature, most studies use random initialization (cf. [3]).

The goal of this paper is to utilize population based algorithms (abbreviated as "PBAs") as initialization booster for NMF. The PBAs are used to initialize the factors W and H in order to minimize the NMF objective function prior to the factorization. The goal is to find a solution with smaller overall error at convergence, and/or to speed up convergence of NMF (i.e., smaller approximation error for a given number of NMF iterations). Instead of initializing the complete factors W and H at once, we sequentially optimize single rows of W and single columns of H, respectively. This allows for parallel/distributed computation by splitting up the initialization into several partly independent sub-tasks. Mathematically, we consider the problem of finding a "good" (ideally the global) solution of an optimization problem with bound constraints in the form:

$$\min_{x \in \Omega} f(x), \tag{2}$$

where $f : \mathbb{R}^N \rightarrow \mathbb{R}$ is a nonlinear function, and Ω is the feasible region. In the context of this paper, f refers to the optimization (i.e., minimization) of the error of a single row or column, respectively, of the NMF approximation $A \approx WH$. Hence, f is usually not convex and may possess many local minima. Since NMF allows only positive or zero values the search space Ω is limited to nonnegative values. In this paper we consider the following optimization algorithms: Genetic algorithms (GA), particle swarm optimization (PSO), fish school search (FSS), differential evolution (DE), and the fireworks algorithm (FWA).

Related work. So far, only few algorithms for non random NMF initialization have been published. [4] used spherical k-means clustering to group column vectors of A as input for W. A similar technique was used in [5]. Another clustering-based method of structured initialization designed to find spatially localized basis images can be found in [6]. [3] used an initialization technique based on two SVD processes called *nonnegative double singular value decomposition* (NNDSVD). Experiments indicate that this method has advantages over the centroid initialization in [4] in terms of faster convergence. In a recent study [7] we have successfully applied feature selection methods for initializing the basis vectors in W. Compared to the methods mentioned before our approach has computational advantages but can only be applied if the class variables of all data objects are available. Summarizing, so far no generally preferable initialization method for NMF exists which motivates for more research in this area.

Only two studies can be found that combine NMF and PBAs, both of them are based on GAs. [8] have investigated the application of GAs on sparse NMF for microarray analysis, while [9] have applied GAs for boolean matrix factorization, a variant of NMF for binary data based on Boolean algebra. The results in these two papers are promising but barely connected to the initialization techniques introduced in this paper. To the best of our knowledge, there are no studies that investigate the application PBAs as initialization enhancers for NMF.

2 Methodology

2.1 The NMF Algorithm

The general structure of NMF algorithms is given in Alg. 1. Usually, W and H are initialized randomly and the whole algorithm is repeated several times (*maxrepetition*). In each repetition, NMF update steps are processed until a maximum number of iterations is reached (*maxiter*). These update steps are algorithm specific and differ from one NMF variant to the other. If the approximation error drops below a pre-defined threshold, or if the shift between two iterations is very small, the algorithm might stop before all iterations are processed.

Given matrix $A \in \mathbb{R}^{m \times n}$ and $k \ll min\{m, n\}$;
for *rep = 1 to maxrepetition* **do**
  $W = \text{rand}(m, k)$;
  $(H = \text{rand}(k, n))$;
  for *i = 1 to maxiter* **do**
    perform algorithm specific NMF update steps;
    check termination criterion;
  end
end

Algorithm 1. General Structure of NMF Algorithms

Algorithmic variants. Several algorithmic variants for computing NMF have been developed. Early algorithms comprise multiplicative update (MU) and alternating least squares (ALS) [1], as well as projected gradient (PG) [10]. Over time, other algorithms were derived, such as a combination of ALS and PG (ALSPGRAD) [10], quasi Newton-type NMF [6], as well as fastNMF and bayesNMF [11].

2.2 Population Based Optimization Algorithms

Genetic Algorithms (GA, [12]) are global search heuristics that operate on a population of solutions using techniques encouraged from evolutionary processes such as mutation, crossover, and selection.

In **Particle Swarm Optimization** (PSO, [13]) each particle in the swarm adjusts its position in the search space based on the best position it has found so far as well as the position of the known best fit particle of the entire swarm.

In **Differential Evolution** (DE, [14]) a particle is moved around in the search-space using simple mathematical formulation, if the new position is an improvement the particles' position is updated, otherwise the new position is discarded.

Fish School Search (FSS, [15, 16]) is based on the behavior of fish schools. The main operators are *feeding* (fish can gain/loose weight, depending on the region they swim in) and *swimming* (mimics the collective movement of all fish).

The **Fireworks Algorithm** (FWA, [17]) is a recently developed swarm intelligence algorithm that simulates the explosion process of fireworks. Two types sparks are generated, based on uniform and Gaussian distribution, respectively.

3 NMF Initialization Using Population Based Algorithms

Before describing new initialization methods using population based algorithms, we discuss some properties of the Frobenius norm (cf. [18]), which is used as objective function to measure the quality of the NMF approximation (Equ. (1)). The Frobenius norm of a matrix $D \in \mathbb{R}^{m \times n}$ is defined as

$$||D||_F = \left(\sum_{i=1}^{min(m,n)} \sigma_i \right)^{1/2} = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |d_{ij}|^2 \right)^{1/2}, \quad (3)$$

where σ_i are the singular values of D, and d_{ij} is the element in the i^{th} row and j^{th} column of D. The Frobenius norm can also be computed row wise or column wise. The *row wise* calculation is

$$||D||_F^{RW} = \left(\sum_{i=1}^{m} |\mathbf{d}_i^r|^2 \right)^{1/2}, \quad (4)$$

where $|\mathbf{d}_i^r|$ is the norm[1] of the i^{th} row vector of D, i.e., $|\mathbf{d}_i^r| = (\sum_{j=1}^{n} |r_j^i|^2)^{1/2}$, and r_j^i is the j^{th} element in row i. The *column wise* calculation is

$$||D||_F^{CW} = \left(\sum_{j=1}^{n} |\mathbf{d}_j^c|^2 \right)^{1/2}, \quad (5)$$

with $|\mathbf{d}_j^c|$ being the norm of the j^{th} column vector of D, i.e., $|\mathbf{d}_j^c| = (\sum_{i=1}^{m} |c_i^j|^2)^{1/2}$, and c_i^j being the i^{th} element in column j. Obviously, a reduction of the Frobenius norm of any row or any column of D leads to a reduction of the total Frobenius norm $||D||_F$. In the following, D refers to the distance matrix of the original data and the approximation, $D = A - WH$.

Initialization procedure. We exploit these properties of the Frobenius norm to initialize the basis vectors in W row wise and the coefficient matrix H column wise. The goal is to find heuristically optimal starting points for single rows of W and single columns of H, which can be computed with all PBAs mentioned in Section 2.2. Alg. 2 shows the pseudo code for the initialization procedure. In the beginning, $H0$ needs to be initialized randomly using a non-negative lower bound for the initialization. In the first loop, W is initialized row wise (cf. Equ. 4), i.e., row $\mathbf{w}_i^r$ is optimized in order to minimize the Frobenius norm of the i^{th} row $\mathbf{d}_i^r$ of D, which is defined as $\mathbf{d}_i^r = \mathbf{a}_i^r - \mathbf{w}_i^r H0$. In the second loop, the columns of H are initialized using on the previously computed rows of W. H is initialized column wise (cf. Equ. 5), i.e., column $\mathbf{h}_j^c$ is optimized in order to minimize the Frobenius norm of the j^{th} column $\mathbf{d}_j^c$ of D, which is defined as $\mathbf{d}_j^c = \mathbf{a}_j^c - W\mathbf{h}_j^c$.

[1] For vectors, the Frobenius norm is equal to the Euclidean norm.

Given matrix $A \in \mathbb{R}^{m \times n}$ and $k \ll min\{m, n\}$;
$H0 = \text{rand}(k, n)$;
for i = *1 to* m **do**
  Use PBAs to find $\mathbf{w}_i^r$ that minimizes $||\mathbf{a}_i^r - \mathbf{w}_i^r H0||_F$, cf. Equ. 4;
  $W(i,:) = \mathbf{w}_i^r$;
end
for j = *1 to* n **do**
  Use PBAs to find $\mathbf{h}_j^c$ that minimizes $||\mathbf{a}_j^c - W\mathbf{h}_j^c||_F$, cf. Equ. 5;
  $H(:,j) = \mathbf{h}_j^c$;
end

Algorithm 2. Pseudo Code for NMF Initialization using PBAs

In line 4, input parameters for the PBAs are $\mathbf{a}_i^r$ (the i^{th} row of A) and $H0$, the output is the initialized row vector $\mathbf{w}_i^r$, the i^{th} row of W. In line 8, input parameters are $\mathbf{a}_j^c$ (the j^{th} column of A) and the *already optimized* factor W, the output is the initialized column vector $\mathbf{h}_j^c$, the j^{th} column of H. Global parameters used for all PBAs are upper/lower bound of the search space and the initialization (the starting values of the PBAs), number of particles (chromosomes, fish, ...), and maximum number of fitness evaluations. The dimension of the optimization problem is identical to the rank k of the NMF.

Parallelism. All iterations within the first *for*-loop and within the second *for*-loop in Algorithm 2 are independent from each other, i.e., the initialization of any row of W does not influence the initialization of any other row of W (identical for columns of H). This allows for a parallel implementation of the proposed initialization method. In the first step, all rows of W can be initialized concurrently. In the second step, the columns of H can be computed in parallel.

4 Experimental Evaluation

For PSO and DE we used the Matlab implementations from [19] and adapted them for our needs For PSO we used the constricted G_{best} topology using the parameters suggested in [20], for DE the crossover probability parameter was set to 0.5. For GA we adapted the Matlab implementation of the continuous genetic algorithm available in the appendix of [21] using a mutation rate of 0.2 and a selection rate of 0.5. For FWA we used the same implementation and parameter settings as in the introductory paper[17], and FSS was self-implemented following the pseudo algorithm and the parameter settings provided in [15]. All results are based on a randomly created, dense 100×100 matrix.

4.1 Initialization Results

At first we evaluate the initial error of the approximation after initializing W and H (i. e., *before* running an NMF algorithm). Figures 1 and 2 show the average approximation error (i. e., fitness of the PBAs) per row and per column, respectively, for a varying number of fitness function evaluations.

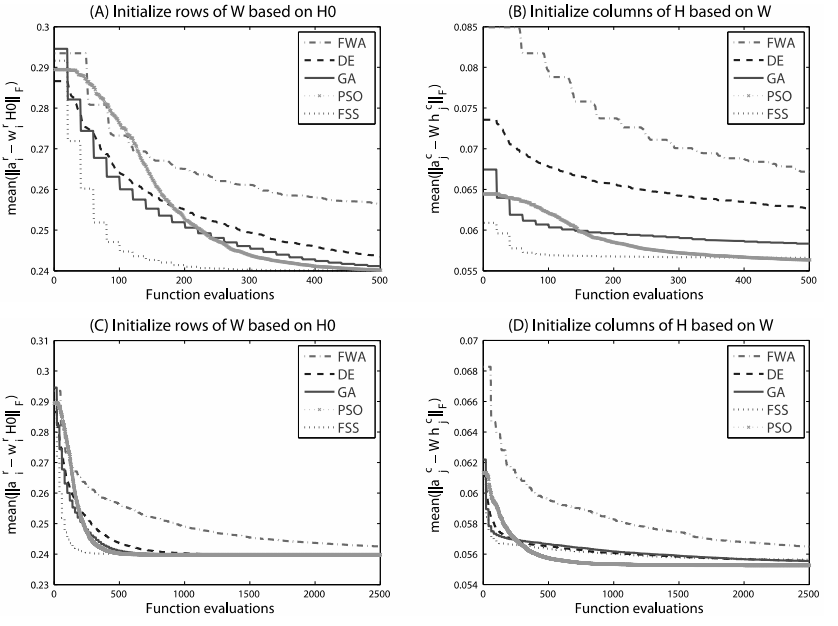

Fig. 1. *Left side:* average appr. error per row (after initializing rows of W). *Right side:* average appr. error per column (after initializing columns of H) – rank k =**5**.

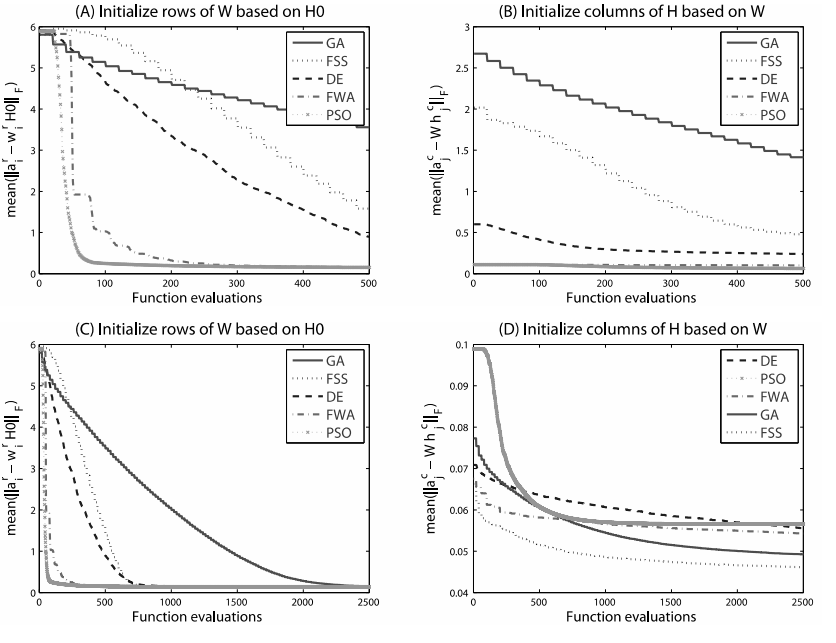

Fig. 2. *Left side:* average appr. error per row (after initializing rows of W). *Right side:* average appr. error per column (after initializing columns of H) – rank k =**30**.

The figures on the left side show the *average* (mean) approximation error per *row* after initializing the rows of W (first loop in Alg. 2). The figures on the right side show the *average* (mean) approximation error per *column* after initializing the columns of H (second loop in Alg. 2). The legends are ordered according to the average approximation error achieved after the maximum number of function evaluations for each figure (top = worst, bottom = best).

Results for k=5. Fig. 1 shows the results achieved for a small NMF rank k set to 5 (k is identical to the problem dimension of the PBAs). In Fig. 1 (A), only 500 evaluations are used to initialize the rows of W based on the randomly initialized matrix $H0$ (see Alg. 2). In Fig. 1 (B) the previously initialized rows of W are used to initialize the columns of H – again using only 500 function evaluations. As can be seen, (to a small amount) GA, DE and especially FWA are sensitive to the small rank k and the small number of function evaluations. PSO and FSS achieve the best approximation results, FSS is the fastest in terms of accuracy per function evaluations. The lower part (C, D) of Fig. 1 shows the results when increasing the number of function evaluations for all PBAs from 500 to 2 500. The first 500 evaluations in (C) are identical to (A), but the results in (D) are different from (B) since they rely on the initialization of the rows of W (the initialization results after the *maximum number* of function evaluations in Fig. 1 (A) and (C) are different). With more function evaluations, all algorithms except FWA achieve almost identical results.

Results for k=30. With increasing complexity (i. e., increasing rank k) FWA clearly improves its results, as shown in Fig. 2. Together with PSO, FWA clearly outperforms the other algorithms when using only 500 function evaluations, see Fig. 2 (A, B). With increasing number of function evaluations, all PBAs achieve identical results when initializing the rows of W (see Fig. 2 (C)). Note that GA needs more than 2 000 evaluations to achieve a low approximation error. When initializing the columns of H (see Fig. 2 (D)), PSO suffers from its high approximation error during the first iterations. The reason for this phenomenon is the relatively sparse factor matrix W computed by PSO. Although PSO is able to reduce the approximation error significantly during the first 500 iterations, the other algorithms achieve slightly better results after 2 500 function evaluations. FSS and GA achieve the best approximation accuracy. The NMF approximation results in Section 4.2 are based on factor matrices W and H initialized with the same parameters as Fig. 2 (C, D): k=30, 2 500 function evaluations.

Parallel implementation. We implemented all population based algorithms in Matlab using Matlab's Parallel Computing Toolbox which allows to run eight workers (threads) concurrently. Compared to sequential execution we achieved a speedup of 7.47, which leads to an efficiency of 0.94. Our current implementation is computationally slightly more demanding than the NNDSVD initialization (cf. Section 1 and 4.2). However, we are currently working on an implementation that allows to use up to 32 Matlab workers (using Matlab's Distributed Computing Server). Since we expect the efficiency to remain stable with increasing number of workers, this implementation should be significantly faster than NNDSVD.

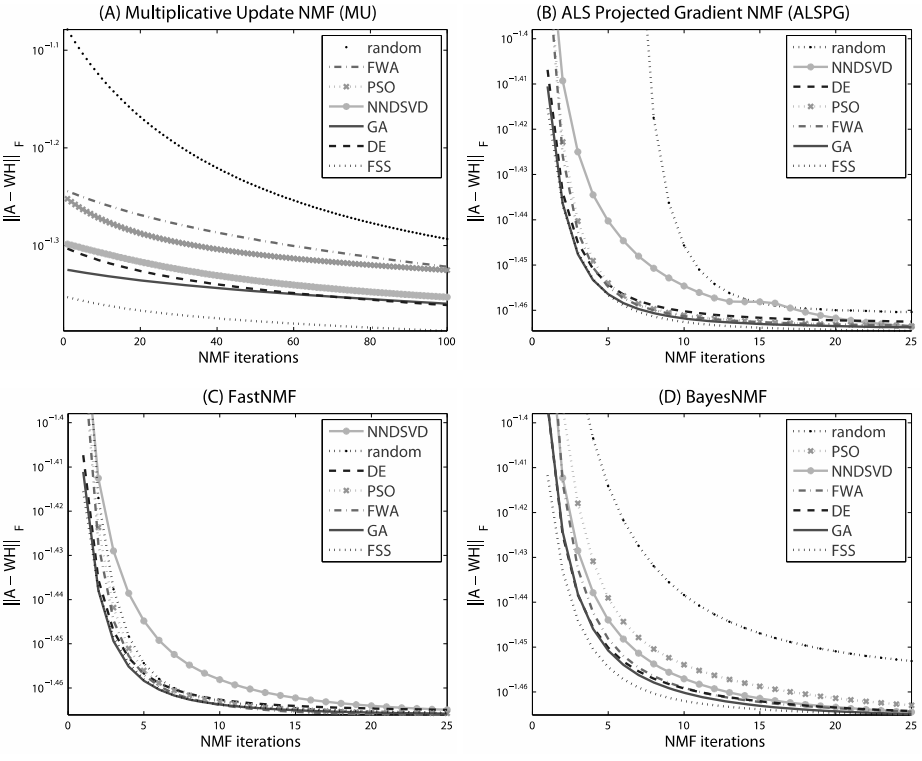

Fig. 3. Approximation error achieved by different NMF algorithms using different initialization variants (k=30, after 2500 fitness evaluations for PBA initialization)

4.2 NMF Approximation Results

In this subsection we report approximation results achieved by NMF using the factors W and H initialized by the PBAs. We compare our results to random initialization as well as to NNDSVD (cf. Section 1), which is the best (in terms of runtime per accuracy) available initialization in the literature for unclassified data. In order to provide reproducible results we used only publicly available Matlab implementations of NMF algorithms. We used the following implementations: Multiplicative Update (MU, implemented in Matlab's Statistics Toolbox), ALS using Projected Gradient (ALSPG, [10]), BayesNMF and FastNMF (both [11]). Matlab code for NNDSVD initialization is also publicly available (cf. [3]).

Fig. 3 shows the approximation error on the y-axis (log scale) after a given number of NMF iterations for four NMF algorithms using different initialization methods. The initialization methods in the legend of Fig. 3 are ordered according to the approximation error achieved after the maximum number of iterations plotted for each figure (top = worst, bottom = best). The classic MU algorithm (A) presented in the first NMF publication [1] has low cost per iteration but converges slowly. Hence, for this algorithm the first 100 iterations are shown. For MU, all initialization variants achieve a smaller approximation error

than random initialization. NNDSVD shows slightly better results than PSO and FWA, but GA, DE and especially FSS are able to achieve a smaller error per iteration than NNDSVD. Since algorithms (B) - (D) in Fig. 3 have faster convergence per iteration than MU but also also have higher cost per iteration, only the first 25 iterations are shown. For ALSPG (B), all new initialization variants based on PBAs are clearly better than random initialization and also achieve a better approximation error than NNDSVD. The performance of the five PBAs is very similar for this algorithm. FastNMF (C) and BayesNMF (D) are two recently developed NMF algorithms which were developed after the NNDSVD initialization. Surprisingly, when using FastNMF, NNDSVD achieves a lower approximation than random initialization, but all initializations based on PBAs are slightly better than random initialization. The approximation error achieved with BayesNMF strongly depends on the initialization of W and H (similar to ALSPG). The PSO initialization shows a slightly higher approximation error that NNDSVD, but all other PBAs are able to achieve a smaller approximation error than the state-of-the-art initialization, NNDSVD.

5 Conclusion

In this paper we introduced new initialization variants for nonnegative matrix factorization (NMF) using five different population based algorithms (PBAs), particle swarm optimization (PSO), genetic algorithms (GA), fish school search (FSS), differential evolution (DE), and fireworks algorithm (FWA). These algorithms were used to initialize the rows of the NMF factor W, and the columns of the other factor H, in order to achieve a smaller approximation error for a given number of iterations. The proposed method allows for parallel implementation in order to reduce the computational cost for the initialization. Overall, the new initialization variants achieve better approximation results than random initialization and state-of-the-art methods. Especially FSS is able to significantly reduce the approximation error of NMF (for all NMF algorithms used), but other heuristics such as DE and GA also achieve very competitive results.

Another contribution of this paper is the comparison of the general applicability of population based algorithms for continuous optimization problems, such as the NMF objective function. Experiments show that all algorithms except PSO are sensitive to the number of fitness evaluations and/or to the complexity of the problem (the problem dimension is defined by the rank of NMF). Moreover, the material provided in Section 4 is the first study that compares the recently developed PBAs, fireworks algorithm and fish school search. Current work includes high performance/distributed initialization, and a detailed comparative study of the proposed methods. A future goal is to improve NMF algorithms by utilizing heuristic search methods to avoid NMF getting stuck in local minima.

Acknowledgments. This work was supported by National Natural Science Foundation of China (NSFC), under Grant No. 60875080. Andreas also wants to thank the *Erasmus Mundus External Cooperation Window*, Lot 14 (agreement no. 2009-1650/001-001-ECW).

References

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

[2] Berry, M.W., Browne, M., Langville, A.N., Pauca, P.V., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization. Computational Statistics & Data Analysis 52(1), 155–173 (2007)

[3] Boutsidis, C., Gallopoulos, E.: SVD based initialization: A head start for nonnegative matrix factorization. Pattern Recogn. 41(4), 1350–1362 (2008)

[4] Wild, S.M., Curry, J.H., Dougherty, A.: Improving non-negative matrix factorizations through structured initialization. Patt. Recog. 37(11), 2217–2232 (2004)

[5] Xue, Y., Tong, C.S., Chen, Y., Chen, W.: Clustering-based initialization for nonnegative matrix factorization. Appl. Math. & Comput. 205(2), 525–536 (2008)

[6] Kim, H., Park, H.: Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J. Matrix Anal. Appl. 30, 713–730 (2008)

[7] Janecek, A.G., Gansterer, W.N.: Utilizing nonnegative matrix factorization for e-mail classification problems. In: Berry, M.W., Kogan, J. (eds.) Survey of Text Mining III: Application and Theory. John Wiley & Sons, Inc., Chichester (2010)

[8] Stadlthanner, K., Lutter, D., Theis, F., et al.: Sparse nonnegative matrix factorization with genetic algorithms for microarray analysis. In: IJCNN 2007: Proceedings of the International Joint Conference on Neural Networks, pp. 294–299 (2007)

[9] Snásel, V., Platos, J., Krömer, P.: Developing genetic algorithms for boolean matrix factorization. In: DATESO 2008 (2008)

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

[11] Schmidt, M.N., Laurberg, H.: Non-negative matrix factorization with Gaussian process priors. Comp. Intelligence and Neuroscience (1), 1–10 (2008)

[12] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman, Amsterdam (1989)

[13] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)

[14] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution A Practical Approach to Global Optimization. Springer, Heidelberg (2005)

[15] Filho, C.J.A.B., de Lima Neto, F.B., Lins, A.J.C.C., Nascimento, A.I.S., Lima, M.P.: Fish school search. In: Chiong, R. (ed.) Nature-Inspired Algorithms for Optimisation. SCI, vol. 193, pp. 261–277. Springer, Heidelberg (2009)

[16] Janecek, A.G., Tan, Y.: Feeding the fish – weight update strategies for the fish school search algorithm. To appear in Proceedings of ICSI 2011: 2nd International Conference on Swarm Intelligence (2011)

[17] Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 355–364. Springer, Heidelberg (2010)

[18] Berry, M.W., Drmac, Z., Jessup, E.R.: Matrices, vector spaces, and information retrieval. SIAM Review 41(2), 335–362 (1999)

[19] Pedersen, M.E.H.: SwarmOps - numerical & heuristic optimization for matlab (2010), `http://www.hvass-labs.org/projects/swarmops/matlab`

[20] Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Swarm Intelligence Symposium, SIS 2007, pp. 120–127. IEEE, Los Alamitos (2007)

[21] Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms, 2nd edn. John Wiley & Sons, Inc., Chichester (2005)