Review of "Positive Matrix Factorization of Large Aerosol Mass Spectrometry Datasets Using Error-Weighted Randomized Hierarchical Alternating Least Squares"

Summary

NMF is a widely recognized technique with many applications in data analytics. For a given data matrix X, the weighted NMF computates nonnegative factors W, H that attempt to minimize the weighted Frobenius error $\|S \circ (X - WH)\|_F^2$ where $\circ$ is the Hadamard product and the elements $(S)_{ij}$ of S are functions of the measurement uncertainties (this is not the authors' notation but see below). The main novelty is the design and incorporation of a form of weighting to existing algorithms for NMF (or PMF, as the authors opt to call them) and its application on atmospheric data. The experimental results indicate that the method is faster but sometimes less accurate than other methods. It is also shown that these (larger) errors are not seriously affecting intepretability. The authors also state that "We will not discuss the chemical interpretations of the data, rather just the mathematical results from running the RHALS algorithm." Therefore, the paper is more on the algorithmic and numerical issues than on the method's impact on the selected application.

RHALS and weighted RHALS have been discussed elsewhere in the literature, so the main novelty appears to be in the weighting scheme and the comparisons. As described, in the external weighting scheme, one computes unweighted factors, e.g. using standard RHALS on the scaled data matrix, then applies the weighting on their product and then computes new nonnegative factors. This latter step is accomplished using alternating least squares. So it seems that the original weighted NMF is solved first after scaling, and then a new NMF is computed. In particular, for this second phase, I assume that some NMF is used and not simply ALS, otherwise the factors will come out of mixed sign. Is this mentioned?. In any case, the idea is interesting but somwewhat ad hoc and a little circular and therefore deserves more discussion. See also the detailed comments for "[page 11], line 273".

The externally weighted algorithm turns out to be faster than existing "internally weighted" NMF methods, specifically MU and HALS. The authors also compare with the MU, ALS and HALS algorithms using the proposed external weighting scheme. A link to a MATLAB implementation is also provided. Overall, the scientific advance in the paper is incremental but can be of interest.

Some points of criticism:

1. The key point of the algorithm is that the combination of RHALS with the external weighting scheme is faster than competing schemes. However, to solidly back this claim, it is not sufficient to only provide runtimes, especially from unoptimized MATLAB implementations. For example, the computational costs of several formulas, e.g the updates on p9, are very sensitive to the order of the operations, the parenthesization, etc. MATLAB is very convenient, but does not attempt to optimize matrix expressions. Were these issues taken into account? The

authors also need to show that the "competing" codes have been implemented with runtime minimization in mind. Unfortunately, it is not sufficient to compare performance-oblivious implementations when the objective is to reduce runtimes.

2. The authors should also specify the costs of each method per iteration (preferably in terms of the matrix primitives used).

3. Comparisons are with weighted versions of ALS and MU. These are prone to slow convergence and or even failure to reach a reasonable local optimal solution. ALS, for example (see N. Gillis, p283 of https://doi.org/10.1137/1.9781611976410) "is easy to implement ... and sometimes provides reasonable solutions, typically for sparse matrices; ... However, it comes with no theoretical guarantee and in fact often diverges in practice, especially for dense input matrices; ... Therefore, ALS can be recommended only as a warm-start stage for theoretically better grounded approaches...." See chapter 8 of the aforementioned book and Table 8.2 for a comparison of methods: ALS and MU stand at the "low end" in several respects.

4. In Section 4.1, I am missing an analysis of the contribution of the number of factors to the cost. Indeed, in some cases the discussion is incomplete: For example on p11 it is stated that ".... the algorithm may take longer for a larger number of factors and in certain cases. More work is needed regarding the different convergence issues that may arise." In light of the fact that the external weighting scheme is a key contribution of this paper, the authors need to address these issues in greater detail.

5. It would also be useful to provide information on the cost of each major step and iteration rather than only comparing runtimes to convergence.

6. The paper makes a brief reference to a WNMF method for missing entries (Yahaya et al.) However the method is dismissed because of the cost of EM without further ado. The authors should address a) how they would handle missing entries, b) how does their method compare in view of related papers by the same team, some included in the publically available thesis of F. Yamaha. It is worth noting that the paper "Random Projection Streams for (Weighted) Nonnegative Matrix Factorization," doi: 10.1109/ICASSP39728.2021.9413496." discusses computational complexity, something I am missing in this manuscript as explained earlier.

7. I spotted some (trivial to correct but somewhat sloppy-type) errors. Some examples: Formula (7) and the formula for the quality of fit parameter (line 161) are incorrect. In the former, one must use the absolute values of the terms of the sum and in the latter it is the individual terms that must be squared, not the sum. Line 324 on p13 also has such an error. On p. 19, line 527, it is stated that "H is a k $\times$ n matrix that is assumed to be of full column rank". However, assuming that $k < n$, it cannot be full column rank, but at best, full row rank. There might be other errors as well.

Detailed comments

[page 1]: Please state whether the matrix is assumed to be strictly positive or nonnegative. If the latter, then why use PMF instead of NMF which is more general and the term used in some of the paper's major citations.

[page 2 and beyond]: The authors state: "In Eq. (1), the division by Σ represents elementwise division". I strongly recommend against using fractional notation when dealing with matrices. They should use Hadamard division or Hadamard multipication with a variable that is defined to contain the inverses of the elements of Σ. In fact, on p265, the authors actually use the (MATLAB infix operator) for elementwise division. This is certainly better than the / symbol, but even better to use the "Hadamard notation".

[page 3]: Better include the section number when you refer to Sections, e.g. Methods (Section 2).

[page 4]: The use of "*" to denote matrix multiplication is inconsistent with the rest of the paper.

[page 4]: A proper and complete discussion on computational optimizations and parallelization would merit much more than the few statements in sections 1.3. I propose that the authors take this into account and eliminate section 1.3.2, incorporating the few comments in some other section.

[page 6]: Please clarify the statement of line 140. How does the incorporation of the weighting via the uncertainty matrix affect the costs?

[page 8]: Formula (10) is missing the opeining (left) parenthesis. Also in line 210 better say "denotes the trace of the matrix" rather than "takes the trace of the matrix". After formula (14) better say "Differentiating with respect..." instead of "Taking the derivative ...".

[page 9]: Careful in the parenthesization of Formula (16). Some parentheses are missing

[page 9]: In formula (20) and elsewhere, aren't the Σ_j matrices diagonal? If so, then the transpose symbol is redundant. Also careful in the parenthesization of Formula (20). Some parethesizations can be more efficient than others.

[page 9]: In formula (20) and elsewhere: have you defined the maximum operator $[\cdot]_+$?

[page 10]: Some readers might question the characterization "rotation" for a non-orthogonal matrix. In the linear algebra literature, rotation matrices preserve the 2-norm and the Frobenius norm and are orthogonal. It might be worth clarifying this.

[page 11]: In line 273, better say "Here $W^\dagger$ and $H^\dagger$ denote the pseudoinverses ..." It is then stated that "Running this code in Matlab on a single CPU, we found the update rules using the pseudoinverses were less computationally expensive." Less expensive than what? How are the pseudoinverses computed? What is the computational cost of this part of the algorithm relative to the rest? How many iterations are necessary? Do they lead to convergence? In particular, looking at the code it seems that the method calls MATLAB's `pinv`. Is this necessary? What if the matrix is $m \times n$ with $n << m$? Also reading function `randomized_nnmf.m` one sees the comment: "NOTE: This code is not adapted from any research papers or pre existing code. The only motivation is that this code seems to produce feasible results and we haven't thought of any better alternatives. More testing needs to be done on the convergence of this method to feasible solutions." Has this been done?

[page 12]: In Figure 3, better say that MATLAB notation is used, in which the digits following the `e` symbol are the exponent to which

[page 13]: there is no nonnegative SVD, the authors mean something else (possibly the so called "nonnegative double SVD" initialization from ref. [1]).

[page 19:] In line 525, $W^\dagger$ must be bold according to the paper's notation. In any case, I suggest not to include Appendix B; its findings are well known linear algebra facts.

[page 21, Bibliography]: Line 543: (Burred) Incomplete reference: no date, publication information, etc.

[Bibliography] The authors should also take into account the WNMF work described in the well cited 2008 PhD KU Louvain thesis by N.D. Ho[1]. Another important reference (as noted earlier) is the book by N. Gillis[2].

[Bibliography] The entry "Tan, W., C. S. F. L. L. C. W. Z. and Cao, L...." is probably wrong. See 10.1145/3225058.3225096.

[page 27]: Fig4: Do you mean $\log n$ or $\log(n^2)$. Which base log? What are the exact sizes of the smallest and largest data sets?

[1] https://perso.uclouvain.be/paul.vandooren/ThesisHo.pdf.
[2] https://doi.org/10.1137/1.9781611976410