A NON-NEGATIVE MATRIX FACTORIZATION GAME

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a novel game-theoretic formulation of Non-Negative Matrix Factorization (NNMF), a popular data-analysis method with many scientific and engineering applications. The game-theoretic formulation is shown to have favorable scaling and parallelization properties, while retaining reconstruction and convergence performance comparable to the traditional Multiplicative Updates (Lee & Seung, 1999) algorithm.

1 INTRODUCTION

Non-Negative Matrix Factorization (NNMF) (Gillis, 2020) is a matrix decomposition method that approximates a low-rank non-negative matrix $\mathbf{X} \in \mathbb{R}_{\geq 0}^{I \times J}$, using two non-negative matrices $\mathbf{W} \in \mathbb{R}_{\geq 0}^{I \times K}$, and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{K \times J}$, of predetermined rank K according to: $\mathbf{X} \simeq \mathbf{WH}$. The non-negative rank K of the non-negative matrix $\mathbf{X}$ can be higher than the traditional matrix rank over the real field.

NNMF has a long history of use in both scientific and engineering settings due to its tendency to reveal interesting properties about the underlying data. For example, Lee & Seung (1999) popularized NNMF after applying it to natural images, showing that it can learn meaningful visual representations "the parts of objects". NNMF has also been applied to large-scale textual corpora to learn topic-models (Lee & Seung, 1999; Pauca et al., 2004) and to high-throughput omics data for time-course analysis (Stein-O'Brien et al., 2018).

Recently Gemp et al. (2020; 2021), reformulated the classic Principal Components Analysis (PCA) Jolliffe (2002) algorithm as a K-player game, to show that it can be be massively scaled and implemented in a distributed manner. Inspired by their work, we present a novel game-theoretic formulation of Non-Negative Matrix Factorization (NNMF) and provide an empirical analysis of our formulation against a traditional NNMF algorithm (Lee & Seung, 1999).

2 BACKGROUND

Principal Components Analysis (PCA) is a data-analysis algorithm that uses an orthogonal linear transformation to transform a set of observations $\mathbf{X} \in \mathbb{R}^{n \times d}$ to a new coordinate system, such that the projections of the data onto the new coordinates have decreasing variance. In other words, the first coordinate (known as the first *principal component*) is the direction of maximal variance of the data, the second (orthogonal) coordinate is the direction of second-highest variance of the data, and so on. (See (Jolliffe, 2002) for a comprehensive review.)

PCAs are typically calculated in one of two ways: (1) using the eigen-decomposition of the covariance matrix Σ of the data matrix $\mathbf{X}$ or (2) using the singular value decomposition (SVD) of the data matrix $\mathbf{X}$. Using the former method, let the data matrix be $\mathbf{X} \in \mathbb{R}^{n \times d}$, and its covariance matrix be $M^{d \times d} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$. PCA then reduces to the eigenvalue problem $MV = V\Lambda$, where Λ is a diagonal matrix comprised of eigenvalues (known as PC *loadings*) and the columns of the orthonormal matrix V provides the sought PC directions.

2.1 GAME-THEORETIC PCA FORMULATION

In a recent work, Gemp et al. (2020) showed how the top-k right singular vectors of an n-observation data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ can be calculated by a k-player game. Alternatively, if one is working with

a mini-batch $\mathbf{X_t}$ of $n' < n$ observations, the top-k right singular vectors are given by the top-k eigenvectors of the positive semidefinite sample covariance matrix $\Sigma = \mathbb{E}[\,\frac{1}{n'}\mathbf{X_t}^T\mathbf{X_t}]$.

In their game formulation, each of k players receives access to a minibatches of the data and 'owns' one of the k (approximate) eigenvectors $\hat{v}_i$. Each player then tries to maximize a utility function $u_i(\hat{v}_i|\hat{v}_{j<i})$ that is unique to them:

$$\max_{\hat{v}_i^\top \hat{v}_i = 1} \left\{ u_i(\hat{v}_i|\hat{v}_{j<i}) = \overbrace{\hat{v}_i^\top \Sigma \hat{v}_i}^{\text{Variance}} - \sum_{j<i} \overbrace{\frac{\langle \hat{v}_i, \Sigma \hat{v}_j \rangle^2}{\langle \hat{v}_j, \Sigma \hat{v}_j \rangle}}^{\perp\text{-penalty}} \right\} \tag{1}$$

Here, all expressions involving the inequality $(j < i)$ indicate that players with a lower index ('parents') have already played before the current ('child') player. As shown above, utility functions consist of two terms that balance competing objectives: one that tries to maximize the variance captured by the player's eigenvector, and the other that tries to maximally orthogonalize this eigenvector to all parent eigenvectors. To reduce underspecification, eigenvectors are restricted to lie on the unit sphere ($\hat{v}_i^{(t)} \in \mathcal{S}^{d-1}$).

Algorithm 1 PCA game (Gemp et al., 2020)

Given: data minibatches $\mathbf{X_t} \in \mathbb{R}^{m \times d}$, total iterations T, initial vector $\hat{v}_i^0 \in \mathcal{S}^{d-1}$ (unit sphere in $d-1$ dimensions), and step size α.
$\hat{v}_i \leftarrow \hat{v}_i^0$
for $t = 1 : T$ **do**
  $\nabla_{\hat{v}_i} \leftarrow 2\mathbf{X_t}^\top \left[\mathbf{X_t}\hat{v}_i - \sum_{j<i} \frac{\langle \mathbf{X_t}\hat{v}_i, \mathbf{X_t}\hat{v}_j \rangle}{\langle \mathbf{X_t}\hat{v}_j, \mathbf{X_t}\hat{v}_j \rangle} \mathbf{X_t}\hat{v}_j \right]$
  $\nabla^R_{\hat{v}_i} \leftarrow \nabla_{\hat{v}_i} - \langle \nabla_{\hat{v}_i}, \hat{v}_i \rangle \hat{v}_i$
  $\hat{v}_i' \leftarrow \hat{v}_i + \alpha \nabla^R_{\hat{v}_i}$
  $\hat{v}_i \leftarrow \hat{v}_i' / ||\hat{v}_i'||$
  `broadcast`($\hat{v}_i$) to all other players
end for
return $\hat{v}_i$

The complete algorithm is given in Algorithm 1. At each iteration, each player updates their eigenvector $\hat{v}_i$ using a fixed step size *Riemannian* gradient ascent update $\nabla^R_{\hat{v}_i}$, that ensures that each iterate lies on the acceptable manifold (Bonnabel, 2013). The different utility functions and order of play results in player-i owning the eigenvector $\hat{v}_i$ associated with the ith largest eigenvalue λ_i. In addition to the above formulation that assumes ordered play, Gemp et al. (2020) empirically show that simultaneous play also converges to Nash equilibrium.

3 NNMF AS A MULTIPLAYER GAME

Before we formulate NNMF as a game, we describe traditional algorithms used for it.

3.1 TRADITIONAL ALGORITHMS FOR NNMF

In its most general form, NNMF is an optimization problem that involves minimizing some loss function $L(\mathbf{X}, \mathbf{WH})$ that penalizes the divergence between the original data matrix $\mathbf{X}$ and its reconstruction $\mathbf{WH}$.

We will work with the following square Frobenius norm penalty for the rest of this manuscript, though more sophisticated objectives, including those that include various regularization terms, are possible (Berry et al., 2007):

$$\underset{w_{i,k}, h_{k,j} \geq 0}{\text{minimize}} ||\mathbf{X} - \mathbf{WH}||_F^2 \tag{2}$$

This optimization formulation is non-convex and obtaining globally optimum solutions is known to be NP-Hard (Vavasis, 2010). However, interest in NNMF has led to the development of many

empirically successful algorithms for finding local optima. These algorithms fall into three general classes (see Berry et al. (2007) for a thorough treatment):

Non-negative Alternating Least squares (NALS): While the optimization in Equation 2 is not convex in both $\mathbf{W}$ and $\mathbf{H}$, it is indeed convex in each term taken individually (*biconvex*). NALS solves a least-squares problem alternatively for $\mathbf{W}$ and $\mathbf{H}$, projecting the iterates into the positive orthant at each iteration, till convergence is observed.

Projected Gradient (PG): Similar to NALS, PG performs a gradient ascent step on $\mathbf{W}$ and $\mathbf{H}$ alternatively, and projects the resulting iterates into the positive orthant at each iteration, till convergence is observed.

Algorithm 2 Multiplicative updates (Lee & Seung, 1999)

1: Given: data matrix, $\mathbf{X} \in \mathbb{R}^{I \times J}$, latent dimension K, maximum iterations T, randomly initialized $\mathbf{W}^0 \in \mathbb{R}_{++}^{I \times K}$ and $\mathbf{H}^0 \in \mathbb{R}_{++}^{K \times J}$.
2: **for** $t = 1 : T$ **do**
3: $\quad \mathbf{H}^{t+1} \leftarrow \mathbf{H}^t \odot \dfrac{((\mathbf{W}^t)^T \mathbf{V})}{((\mathbf{W}^t)^T \mathbf{W}^t \mathbf{H}^t)}$
4: $\quad \mathbf{W}^{t+1} \leftarrow \mathbf{W}^t \odot \dfrac{(\mathbf{V}(\mathbf{H}^{t+1})^T)}{(\mathbf{W}^t \mathbf{H}^{t+1} (\mathbf{H}^{t+1})^T)}$
5: **end for**
6: return $\mathbf{W}^t$ and $\mathbf{H}^t$
Note: $\odot$ and $\mathbf{X}/\mathbf{Y}$ are elementwise multiplication and division respectively.

Multiplicative updates (MU): In PG, the choice of gradient step size is typically done heuristically. As an improvement, (Lee & Seung, 1999) popularized a variant of PG that uses an adaptive step size and results in a multiplicative update for $\mathbf{W}$ and $\mathbf{H}$ at each iteration. The complete algorithm is given in Algorithm 2.

3.2 Game-theoretic NNMF formulation

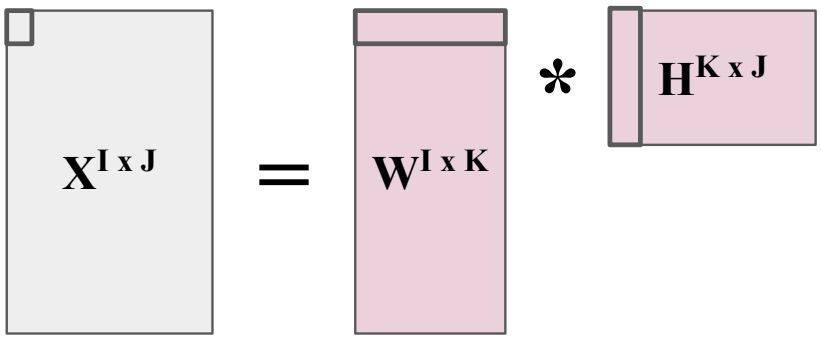

Figure 1: **NNMF as a multiplayer game**: NNMF is formulated as a game between I row-players, each owning one row of the $\mathbf{W}$ matrix, that play with J column-players, each owning one column of the $\mathbf{H}$ matrix. In each interaction, one row-player i and one column-player j maximize their respective utilities to better reconstruct one element of the data-matrix $\mathbf{X}_{[i,j]}$. Additional games between the column players can be introduced to nudge the game towards desirable solutions, such as orthogonal rows for $\mathbf{H}$.

To set up NNMF as a game, we define two sets of players, I row-players and J column-players and their respective utilities. Each of the I row-players owns a row $\mathbf{W}_{[i,:]}$ of the $\mathbf{W}$ matrix, and each the J column-players owns a column $\mathbf{H}_{[:,j]}$ of the $\mathbf{H}$ matrix. Therefore, each player owns a length-K non-negative vector.

Taking inspiration from the game formulation of PCA (Gemp et al., 2020), we now define a utility function for each player, such that a player's local actions (i.e. best response) achieve a global optimization objective. The primary interaction in this game is between row-players playing column-players and vice-versa. The utility function in these interactions is given by:

$$u_{(i,j)} = -\ell(i,j) = -\frac{1}{2}[\,\mathbf{X}_{[i,j]} - (\mathbf{W}_{[i,:]})^T \mathbf{H}_{[:,j]}]^2 \tag{3}$$

In other words, the ith row-player and j^{th} column-player *cooperate* to try to reconstruct the $(i,j)^{th}$ element of the data matrix $\mathbf{X}$. The NNMF game algorithm is summarized in Algorithm 3.

The formulation described up to this point can be regarded as a *graphical game* on a bipartite graph with row-players and column players comprising the two disjoint sets. We now relax the bipartite structure to a more general graphical game where there are interactions between some identical players, similar to replicator or winner-take-all dynamics from evolutionary game theory (Sandholm, 2010). Column players (owning columns of the $\mathbf{H}$ matrix) now have *self-games* as a way to induce certain forms of regularization that aid interpretation of the resulting matrix $\mathbf{H}$. For example, to encourage non-overlapping rows in $\mathbf{H}$, we could have a column player play against itself in such a way that the smallest element of the column-vector it owns is set to zero. A more aggressive regularization strategy towards the same goal could be that all but the largest element of the column-vector stay non-zero after the self-interaction. Henceforth, we call these strategies `J-min` and `J-max` respectively, In practice by instead of setting an element of the column-vector to zero, we multiply it by 0.99 to nudge it in the right direction.

Algorithm 3 NNMF game (simultaneous play)

Given: data matrix, $\mathbf{X} \in \mathbb{R}^{I \times J}$, latent dimension K, maximum iterations T, step size η, randomly initialized $\mathbf{W}^0 \in \mathbb{R}_{++}^{I \times K}$ and $\mathbf{H}^0 \in \mathbb{R}_{++}^{K \times J}$.
for $t = 1 : T$ **do**
  for $i = 1 : I$ **do**
    for $j = 1 : J$ **do**
      $\nabla_W^t \leftarrow -[\,\mathbf{X}_{[i,j]} - (\mathbf{W}_{[i,:]}^t)^T \mathbf{H}_{[:,j]}^t]\,(\mathbf{H}_{[:,j]}^t)^T$
      $\nabla_H^t \leftarrow -[\,\mathbf{X}_{[i,j]} - (\mathbf{W}_{[i,:]}^t)^T \mathbf{H}_{[:,j]}^t]\,(\mathbf{W}_{[i,:]}^t)^T$
      $\mathbf{W}_{[i,:]}^{t+1} \leftarrow max[\,\mathbf{0}, \mathbf{W}_{[i,:]}^t - \eta \nabla_W^t]$
      $\mathbf{H}_{[:,j]}^{t+1} \leftarrow max[\,\mathbf{0}, \mathbf{H}_{[:,j]}^t - \eta \nabla_H^t]$
    end for
  end for
end for
return $\mathbf{W}^t$ and $\mathbf{H}^t$

4 Analysis

4.1 Empirical analysis:

To validate our new game-theoretic formulation of NNMF, we empirically compare it with the traditional Multiplicative Updates (MU) algorithm using synthetic datasets.

Synthetic dataset generator: For all experiments, we set the number and dimension of our observations to be $I = 100$ and $J = 20$ respectively, resulting in a data matrix $\mathbf{X_{syn}} \in \mathbb{R}^{I \times J}$. We set the true number of latent factors to $K = 3$, and generate a random Uniform$(0,1)$ non-negative mixing matrix $\mathbf{W_{syn}} \in \mathbb{R}_{++}^{I \times K}$. We generate the non-negative basis matrix $\mathbf{H_{syn}} \in \mathbb{R}_{++}^{K \times J}$ by applying a Gaussian filter (window-length=3) to the rows of a $K \times J$ random matrix with elements drawn from a Uniform$(0,1)$ distribution. $\mathbf{X_{syn}}$ is then obtained multiplying $\mathbf{W_{syn}}$ and $\mathbf{H_{syn}}$ (Figure 2).

Experiment: We generate a synthetic dataset and compare its decomposition using MU algorithm to the one obtained using our game-theoretic NNMF algorithm and its two regularized variants, `J-min`

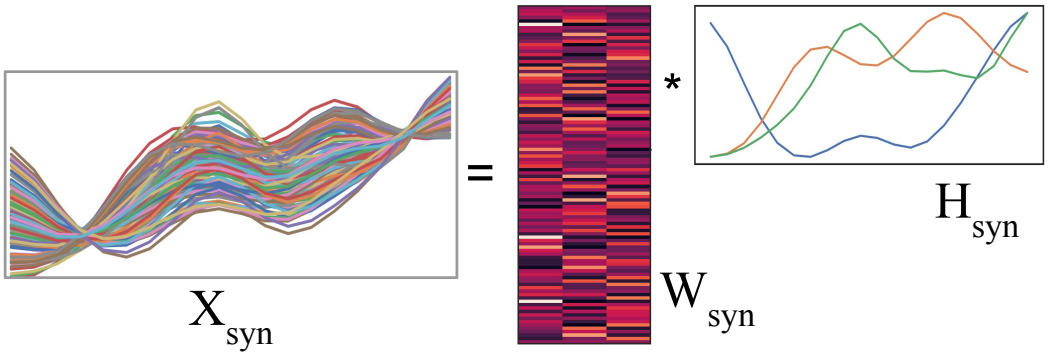

Figure 2: Synthetic data for experiments: Data matrix $\mathbf{X_{syn}} \in \mathbb{R}^{I \times J}$ reconstructed from matrix multiplication of $\mathbf{H_{syn}} \in \mathbb{R}_{++}^{K \times J}$ and $\mathbf{W_{syn}} \in \mathbb{R}_{++}^{I \times K}$.

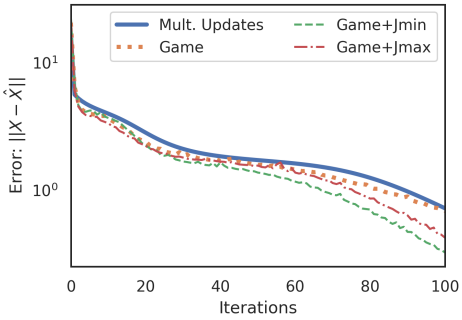

Figure 3: **Convergence of algorithm variants**: Game-theoretic variants of NNMF ($\eta = 0.001$) have convergence rates comparable to the traditional Multiplicative Updates (MU) algorithm.

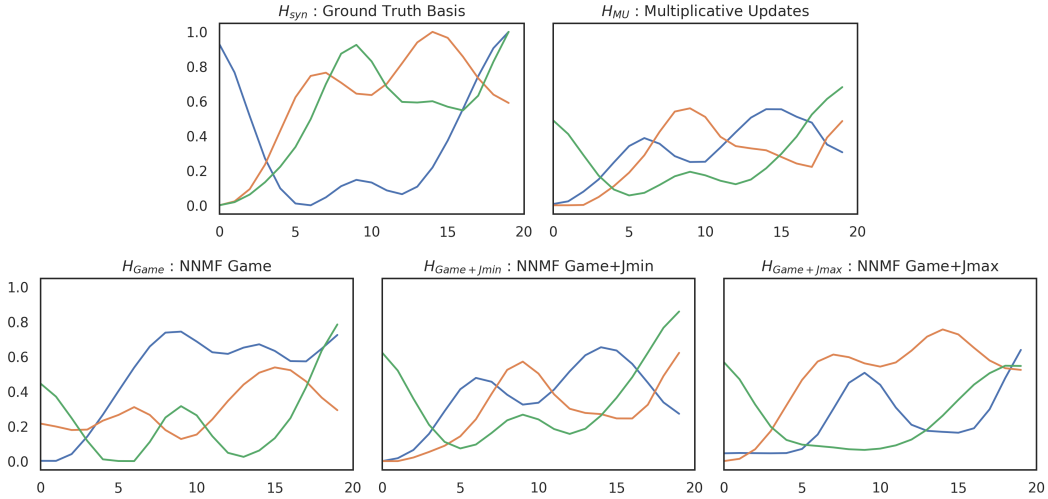

Figure 4: **Ground truth vs. Recovered Bases**: Basis matrices $\mathbf{H_{syn}}$ recovered from MU and Game variants look similar, up to permutations of order (colors). The `J-min` and `J-max` variants only show slight effects of regularization, possibly because the ground-truth basis is not exactly orthogonal.

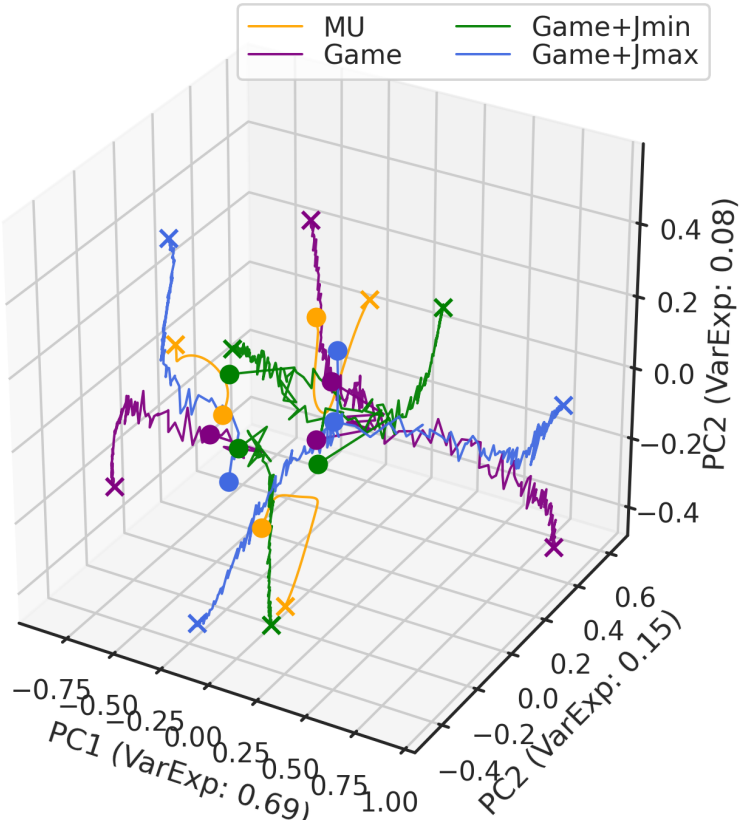

Figure 5: **Iterate trajectories across algorithms show signatures of congestion, but vary in smoothness:** Trajectories for Multiplicative Updates (MU) [orange], NNMF game vanilla [purple], `Jmin` [green], and `Jmax` [blue] variants (using same initialization as in Figure 4). Each algorithm has $K = 3$ trajectories associated with it, one for each row of the recovered basis matrix $\hat{\mathbf{H}}_{\mathbf{syn}}$, each starting at a $\bullet$ marker and ending at a $\times$ marker. Trajectories are observed to be relatively smooth for the (adaptive step-size) MU algorithm and relatively jagged for the (fixed step-size) NNMF game variants. For any algorithm, iterates across trajectories seem to moving away from each other, which might be a signature of congestion (similar to that for the PCA game (Gemp et al., 2020)). Low dimensional space is generated using common Principal Component (PC) space obtained from PCA of all $\hat{\mathbf{H}}_{\mathbf{syn}}$ basis vectors from all algorithms stacked together. PC1, PC2 and PC3 explain 69, 15 and 8 percent of the total variance of the data respectively.

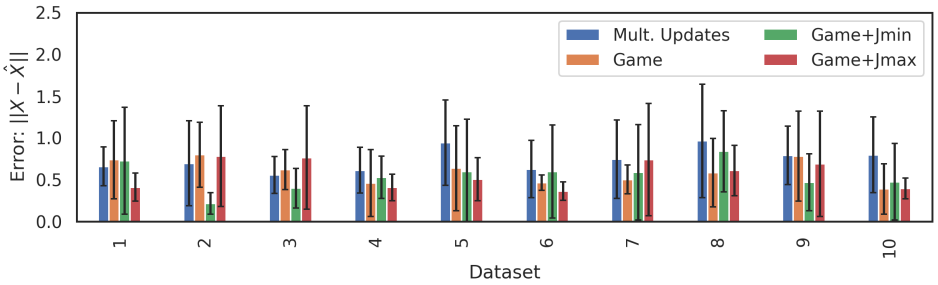

Figure 6: **Reconstruction error of algorithm variants evaluated over multiple randomly generated synthetic datasets**: Game-theoretic NNMF variants result in reconstruction errors that are not consistently significantly different compared to reconstruction errors from the traditional MU algorithm. Error bars represent $\pm$ 1 s. d., calculated over 4 random initializations.

and `J-max`, Section 3.2 (using a fixed gradient step size of $\eta = 0.001$ across all NNMF executions). We find that the convergence trajectories of the MU and NNMF-Game algorithms are comparable, and that the recovered basis matrices $\hat{\mathbf{H}}_{syn}$ look quite similar (Figure 3). Next, we examine the evolution of the non-negative basis (row) vectors that comprise the basis matrix $\hat{\mathbf{H}}_{syn}$ by plotting a low-dimensional projection of the trajectories of the iterates (Figure 5). This reveals trajectories that appear smooth for the (adaptive step size) MU algorithm and relatively jagged for our (fixed step size) NNMF game variants. Finally, we compare the reconstruction performance ($Error = ||\mathbf{X} - \hat{\mathbf{X}}||_2$) of these algorithms across 10 new independently randomly generated synthetic datasets, with 4 randomly initialized episodes for each. We find that the performance of the game-theoretic NNMF variants and that of the traditional MU algorithm is not significantly different (Figure 6).

4.2 Parallelization and scaling properties

The game-theoretic formulation of NNMF has several favorable parallelization and scaling properties:

Data sharding: Each row-player and each column-player only require access to their respective row or column of the data matrix $\mathbf{X}$. In Algorithm 3 only a single element of the data matrix is required for every row-column player interaction. If each row-player and each column-player make a local copy of their associated row or column of the data-matrix, that would be sufficient for all possible row-column interactions they have. Furthermore, if we assume ordered play in row-column interactions, only the leader need hold the data matrix. They can then share the required ground-truth matrix element with the other player during the interaction. In our formulation, we assume that each (row or column) player is on a separate computational core. However, the algorithm is indifferent to the actual distribution of players across computational cores. This distribution can be arbitrary or be dictated by some optimal trade-off between inter-core communication cost and per-core memory cost.

Computational cost: The computational cost, in floating point operations (FLOPS), of multiplying one $I \times K$ matrix with a $K \times J$ matrix is $O(IJK)$. Therefore one MU iteration (Algo. 2) costs $O(IJK)$ $[((\mathbf{W}^t)^T\mathbf{V})]$ + $O(IK^2 + JK^2)$ $[((\mathbf{W}^t)^T\mathbf{W}^t\mathbf{H}^t)]$ + $O(KJ)$ $[\mathbf{X}/\mathbf{Y}]$ + $O(KJ)$ $[\odot]$ for the $\mathbf{H}$ update. A similar calculation applies to the $\mathbf{W}$ update.

One NNMF iteration (Algorithm 3) involves IJ row-column player interactions, each with a cost $O(K^2)$ $[\nabla^t_W \leftarrow -[\,\mathbf{X}_{[i,j]} - (\mathbf{W}^t_{[i,:]})^T\mathbf{H}^t_{[:,j]}]\,(\mathbf{H}^t_{[:,j]})^T]$ + $O(K)$ $[\ \mathbf{W}^{t+1}_{[i,:]} \leftarrow \mathbf{W}^t_{[i,:]} - \eta\nabla^t_W\ ]$ for the $W_{[i,:]}$ update. A similar calculation applies for the $\mathbf{H}$ update. This is comparable with the cost of the MU update for small K, while being highly parallelizable.

5 Discussion

We present a game formulation of the popular Non-negative Matrix Factorization algorithm with favorable parallelization and scaling properties. The game structure admits variants that can induce certain forms of interpretability-enhancing regularization such as encouraging rows in the recovered basis matrix $\hat{\mathbf{H}}$ to be orthogonal. We provide empirical evidence that our algorithms have convergence and reconstruction performance similar to those of the traditional Multiplicative Updates (MU) (Lee & Seung, 1999) algorithm. Visual inspection of convergence trajectories for rows of the recovered basis matrix $\hat{\mathbf{H}}$ reveals that iterates tend to vary relatively smoothly for the (adaptive step-size) MU algorithm but relatively jaggedly for our (fixed step-size) NNMF game variants.

For future work, we plan to formally analyse the convergence properties of the graphical game variants introduced in this paper, and explore adaptive learning rate and additional regularization strategies that could result in smoother convergence trajectories and additional desirable properties in the recovered factor matrices. Our approach is related to the all-at-once formulation of NNMF (Acar et al., 2011; Flores Garrido, 2008) and could benefit from more recent work on accelerated iterative algorithms for NNMF (Gillis & Glineur, 2012; Ang & Gillis, 2019). Exploring extensions to the more general Tensor Factorization (Cichocki et al., 2007) setting, and connections to Message Passing algorithms (Kim et al., 2010) are other directions that seem promising.

REFERENCES

Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422*, 2011.

Andersen Man Shun Ang and Nicolas Gillis. Accelerating nonnegative matrix factorization algorithms using extrapolation. *Neural computation*, 31(2):417–439, 2019.

Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

Silvere Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. Nonnegative matrix and tensor factorization [lecture notes]. *IEEE signal processing magazine*, 25(1):142–145, 2007.

Marisol Flores Garrido. *An all-at-once approach to nonnegative tensor factorizations*. PhD thesis, University of British Columbia, 2008.

Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. Eigengame: PCA as a Nash Equilibrium. *arXiv preprint arXiv:2010.00554*, 2020.

Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. Eigengame unloaded: When playing games is better than optimizing. *arXiv preprint arXiv:2102.04152*, 2021.

Nicolas Gillis. *Nonnegative matrix factorization*. SIAM, 2020.

Nicolas Gillis and François Glineur. Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. *Neural computation*, 24(4):1085–1105, 2012.

Ian T Jolliffe. Principal components in regression analysis. In *Principal Component Analysis*. Springer, 2002.

Byung-Hak Kim, Arvind Yedla, and Henry D Pfister. Imp: A message-passing algorithm for matrix completion. In *2010 6th International Symposium on Turbo Codes & Iterative Information Processing*, pp. 462–466. IEEE, 2010.

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

V Paul Pauca, Farial Shahnaz, Michael W Berry, and Robert J Plemmons. Text mining using non-negative matrix factorizations. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pp. 452–456. SIAM, 2004.

William H Sandholm. *Population games and evolutionary dynamics*. MIT press, 2010.

Genevieve L Stein-O'Brien, Raman Arora, Aedin C Culhane, Alexander V Favorov, Lana X Garmire, Casey S Greene, Loyal A Goff, Yifeng Li, Aloune Ngom, Michael F Ochs, et al. Enter the matrix: factorization uncovers knowledge from omics. *Trends in Genetics*, 34(10):790–805, 2018.

Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2010.