Interpretable Matrix Factorization with Stochasticity Constrained Nonnegative DEDICOM

Rafet Sifa[1,2], Cesar Ojeda[1], Kostadin Cvejoski[1], and Christian Bauckhage[1,2]

[1] Fraunhofer IAIS, Sankt Augustin, Germany
[2] University of Bonn, Bonn, Germany
www.multimedia-pattern-recognition.info
{name.surname}@iais.fraunhofer.de

Abstract. Decomposition into Directed Components (DEDICOM) is a special matrix factorization technique to factorize a given asymmetric similarity matrix into a combination of a loading matrix describing the latent structures in the data and an asymmetric affinity matrix encoding the relationships between the found latent structures. Finding DEDICOM factors can be cast as a matrix norm minimization problem that requires alternating least square updates to find appropriate factors. Yet, due to the way DEDICOM reconstructs the data, unconstrained factors might yield results that are difficult to interpret. In this paper we derive a projection-free gradient descent based alternating least squares algorithm to calculate constrained DEDICOM factors. Our algorithm constrains the loading matrix to be column-stochastic and the affinity matrix to be nonnegative for more interpretable low rank representations. Additionally, unlike most of the available approximate solutions for finding the loading matrix, our approach takes the entire occurrences of the loading matrix into account to assure convergence. We evaluate our algorithm on a behavioral dataset containing pairwise asymmetric associations between variety of game titles from an online platform.

Keywords: Unsupervised Learning, Matrix Factorization, Constrained Optimization, Behavior Analysis

1 Introduction

Matrix and tensor factorization methods have been widely used in data science applications for mostly understanding hidden patterns in the datasets and learning representations for variety of prediction tasks and recommender systems [1, 5, 10, 11, 13, 16, 17]. Decomposition into Directed Components (DEDICOM) [8] is a matrix and tensor factorization technique and a compact way of finding low rank representations from asymmetric similarity data. The method has found applications in various data science problems including analysis of temporal graphs [1], first order customer migration [15], natural language processing [4], spatio-temporal representation learning [2, 16] and link prediction in

Copyright © 2017 by the paper's authors. Copying permitted only for private and academic purposes.
In: M. Leyer (Ed.): Proceedings of the LWDA 2017 Workshops: KDML, FGWM, IR, and FGDB.
Rostock, Germany, 11.-13. September 2017, published at http://ceur-ws.org

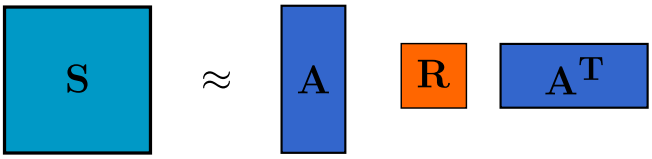

Fig. 1: Illustration of two-way DEDICOM to partition a given similarity matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ encoding pair-wise asymmetric relations among n objects into a combination of a loading matrix $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and a square affinity matrix $\boldsymbol{R} \in \mathbb{R}^{k \times k}$. In this representation $\boldsymbol{A}$ contains the latent factors where the columns describe the hidden structures in $\boldsymbol{S}$ and the relations among these structures are encoded by the asymmetric affinity matrix $\boldsymbol{R}$. Variety of constrained versions of DEDICOM has been previously studied so as to improve the interpretability and the performance of finding appropriate factors.

relational and knowledge graphs [13], where the source of the addressed problems were primarily about analyzing asymmetric relationships between predefined entities.

Formally, given an asymmetric similarity matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ encoding pair-wise asymmetric relations (i.e. $s_{ij} \neq s_{ji}$) among n objects, two-way DEDICOM finds a loading matrix $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and a square affinity matrix $\boldsymbol{R} \in \mathbb{R}^{k \times k}$ for representing $\boldsymbol{S}$ as

$$\boldsymbol{S} \approx \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T. \tag{1}$$

More precisely, DEDICOM approximation of the asymmetric association between elements i and j (i.e. s_{ij}) can be formulated in column vector notation as

$$s_{ij} \approx \boldsymbol{a}_{i:}^T \boldsymbol{R} \boldsymbol{a}_{\boldsymbol{j}:} = \sum_{b=1}^{n} \left(a_{ib} \boldsymbol{r}_{\boldsymbol{b}:}\right)^T \boldsymbol{a}_{\boldsymbol{j}:} = \sum_{b=1}^{n} \sum_{c=1}^{n} a_{ib} r_{bc} a_{jc}, \tag{2}$$

where $\boldsymbol{a}_{\boldsymbol{i}:}$ and $\boldsymbol{a}_{\boldsymbol{j}:}$ represent the ith and jth row of $\boldsymbol{A}$ respectively and $\boldsymbol{r}_{\boldsymbol{b}:}$ represents the bth row of $\boldsymbol{R}$. In this representation $\boldsymbol{A}$ contains the latent factors where the columns describe the hidden structures in $\boldsymbol{S}$ and the relations among these structures are encoded by the asymmetric affinity matrix $\boldsymbol{R}$. A pictorial illustration of two-way DEDICOM factorization is shown in Fig. 1.

Considering the three factor approximations of the similarity values as in (1) and (2), the interpretation (especially with respect to scale) of the hidden structures from given factor matrices $\boldsymbol{A}$ and $\boldsymbol{R}$ might yield misleading results with the presence of negative values [15]. That is, the interpretation of the results is limited to only consider nonnegative affinities in $\boldsymbol{R}$ or the positive or negative loadings of the corresponding points in (2). Additionally, when analyzing nonnegative data matrices (such as ones containing probabilities, counts and etc.), it is usually beneficial to consider nonnegative factor matrices that can be considered as condensed (or compressed) representations that can be used for informed decision making and representation learning [1, 8, 15, 16].

In this work we address the issue of interpretability of the DEDICOM factors by introducing a converging algorithm as an alternative to the previously proposed methods in [1,13,15,16]. Our method constrains the loading matrix $\boldsymbol{A}$ to contain column stochastic vectors and (similar to [15,16]) the affinity matrix $\boldsymbol{R}$ to be nonnegative. Formally this amounts to factorize a given asymmetric similarity matrix $\boldsymbol{S}$ as in (1) by enforcing

$$r_{ij} \geq 0 \; \forall \; \{i, j\} \tag{3}$$

and

$$a_{cb} \geq 0 \wedge \sum_{q=1}^{n} a_{qb} = 1 \; \forall \; \{c, b\}. \tag{4}$$

Compared to the additive-scaling based representation introduced in [15,16], constraining the columns of $\boldsymbol{A}$ to contain probabilities has the direct advantage of interpreting each element of $\boldsymbol{A}$ as the *representativeness* value of the particular loading it represents. That is, the value of a_{ib} represents how much the ith element in the dataset contributes to the bth latent factor. This is indeed parallel to the idea of Archetypal Analysis [3,6,17], where the mixture matrices respectively determine how much a data point and archetypes contribute to respectively constructing the archetypes and reconstructing each data point using the archetypes.

In the following we will first describe the general alternating least squares optimization framework and give an overview of the algorithms to find appropriate constrained and unconstrained DEDICOM factors. After that we will introduce our algorithm by first studying our problem setting and deriving the algorithm step-by-step. This will be followed by a case study covering a real world application where we will analyze game-ownership patterns from data containing asymmetric associations between games using the factors we extract by running our algorithm. Finally, we will conclude our paper with a summary and some directions for future work.

2 Algorithms for Finding DEDICOM Factors

Finding appropriate DEDICOM factors for matrix decomposition can be cast as a matrix norm minimization problem with the objective

$$E(\boldsymbol{A}, \boldsymbol{R}) = \left\| \boldsymbol{S} - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T \right\|^2, \tag{5}$$

which is minimized with respect to $\boldsymbol{A}$ and $\boldsymbol{R}$. Non-convex minimization problems of this kind usually follow an iterative alternating least squares (ALS) procedure where at each iteration we optimize over a selected factor treating the other factors fixed. It is important to note that, the minimized objective function in (5) is convex in $\boldsymbol{R}$ for fixed $\boldsymbol{A}$ whereas not convex in $\boldsymbol{A}$ for fixed $\boldsymbol{R}$, which leads to optimal and sub-optimal approximate solutions when respectively updating $\boldsymbol{R}$ and $\boldsymbol{A}$.

Starting with defining update rules for the affinity matrix $\boldsymbol{R}$, we note that with fixed $\boldsymbol{A}$, minimization of (5) is a matrix regression problem [1, 13, 15] with global optimum solution

$$\boldsymbol{R} \leftarrow \boldsymbol{A}^{\dagger}\boldsymbol{S}\boldsymbol{A}^{T\dagger}, \tag{6}$$

where $\boldsymbol{A}^{\dagger}$ indicates Moore-Penrose pseudoinverse of $\boldsymbol{A}$. Furthermore, if $\boldsymbol{A}$ is constrained to be column orthogonal (i.e. $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_k$, where $\boldsymbol{I}_k$ is the $k \times k$ identity matrix), the update for $\boldsymbol{R}$ simplifies to $\boldsymbol{R} \leftarrow \boldsymbol{A}^T\boldsymbol{S}\boldsymbol{A}$ [15]. Here the updates are not constrained to fall into a particular class of domain and might result in with affinity matrices containing negative elements. Constraining $\boldsymbol{R}$ when minimizing (5) to contain only nonnegative values can be cast as a nonnegative least squares problem by transforming the arguments of (5) as

$$E(\boldsymbol{R}) = \left\| \mathrm{vec}(\boldsymbol{S}) - \mathrm{vec}(\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T) \right\|^2 \tag{7}$$

where the vec operator vectorizes (or flattens) a matrix $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ as

$$\mathrm{vec}(\boldsymbol{B}) = [b_{11},\ \ldots,\ b_{m1},\ \ldots,\ b_{1n},\ \ldots,\ b_{mn}]^T. \tag{8}$$

It is worth mentioning that since vectorizing (5) as shown in (7) does not affect the value of the norm [15], we can make use of the property of vectorization to represent a vectorization of multiplications of matrices as matrix vector multiplication to rewrite (7) as

$$E(\boldsymbol{R}) = \left\| \mathrm{vec}(\boldsymbol{S}) - \Big(\boldsymbol{A} \otimes \boldsymbol{A}\Big)\, \mathrm{vec}(\boldsymbol{R})) \right\|^2, \tag{9}$$

where $\otimes$ represents the Kronecker product. As noted in [15], the expression in (9) can indeed be mapped to constrained linear regression problem [12] with a converging result (through an active set algorithm) to define an update as

$$\boldsymbol{R} \leftarrow \underset{\boldsymbol{R}}{\mathrm{argmin}} \left\| \mathrm{vec}(\boldsymbol{S}) - \Big(\boldsymbol{A} \otimes \boldsymbol{A}\Big)\, \mathrm{vec}(\boldsymbol{R})) \right\|^2 \ \wedge\ r_{ij} \geq 0\ \forall\ i, j. \tag{10}$$

Having identified alternating least squares updates for constrained and unconstrained affinity matrix $\boldsymbol{R}$, we now turn our attention to define updates for the loading matrix $\boldsymbol{A}$. To this end, there have been two different directions followed previously: approximating the minimization of (5) by treating a particular factor with $\boldsymbol{A}$ constant [1, 13] and directly minimizing (5) by means of gradient based approaches. To start with, former method considers stacking matrix $\boldsymbol{S}$ and matrix $\boldsymbol{S}^T$ as

$$\Big[\boldsymbol{S}\ \boldsymbol{S}^T\Big] = \Big[\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\ \boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\Big] = \boldsymbol{A}\Big[\boldsymbol{R}\boldsymbol{A}^T\ \boldsymbol{R}^T\boldsymbol{A}^T\Big], \tag{11}$$

and solves for $\boldsymbol{A}$ keeping $\boldsymbol{A}^T$ fixed [1, 13] to come up with an update for $\boldsymbol{A}$ as

$$\boldsymbol{A} \leftarrow \Big(\boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^T + \boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R}\Big)\Big(\boldsymbol{R}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}^T + \boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\Big)^{-1}. \tag{12}$$

The latter method's ALS update for the loading matrix $\boldsymbol{A}$ is based on directly minimizing (5) by considering first order gradient based optimization [15, 16], which in each update moves the current solution for $\boldsymbol{A}$ in the opposite direction of gradient of (5). To derive the gradient matrix we start by defining (5) in terms of the trace operator as

$$\begin{aligned} E(\boldsymbol{A}, \boldsymbol{R}) &= \operatorname{tr}\left[\boldsymbol{S}^T\boldsymbol{S} - \boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T - \boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{S} + \boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\right] \\ &= \operatorname{tr}\left[\boldsymbol{S}^T\boldsymbol{S} - 2\boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T + \boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\right] \\ &= \operatorname{tr}\left[\boldsymbol{S}^T\boldsymbol{S}\right] - 2\operatorname{tr}\left[\boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\right] + \operatorname{tr}\left[\boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\right]. \end{aligned} \tag{13}$$

Since traces are linear mappings and the term $\operatorname{tr}\left[\boldsymbol{S}^T\boldsymbol{S}\right]$ does not depend on $\boldsymbol{A}$, minimizing $E(\boldsymbol{A}, \boldsymbol{R})$ in (13) with respect to $\boldsymbol{A}$ is equivalent to minimizing $E(\boldsymbol{A})$ which we define as

$$E(\boldsymbol{A}) = E_1(\boldsymbol{A}) + E_2(\boldsymbol{A}) \tag{14}$$

given that

$$E_1(\boldsymbol{A}) = -2\operatorname{tr}\left[\boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\right] \tag{15}$$

and

$$E_2(\boldsymbol{A}) = \operatorname{tr}\left[\boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\right]. \tag{16}$$

Considering the orthogonality constraint on $\boldsymbol{A}$, the term in (16) becomes independent of $\boldsymbol{A}$. That is, since traces are invariant under cyclic permutation we can reformulate (16) as

$$E_2(\boldsymbol{A}) = \operatorname{tr}\left[\boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\right] = \operatorname{tr}\left[\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\boldsymbol{A}\right] = \operatorname{tr}\left[\boldsymbol{R}^T\boldsymbol{R}\right], \tag{17}$$

which allows for only considering the minimization of the error term in (15) [15, 16]. Consequently, we can define a gradient descent based update by considering the gradient matrix

$$\frac{\partial E_1(\boldsymbol{A})}{\partial \boldsymbol{A}} = -2\Big(\boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R} + \boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^T\Big), \tag{18}$$

which with a learning rate $\eta_{\boldsymbol{A}}$ allows us to define an update for $\boldsymbol{A}$ as

$$\boldsymbol{A} \leftarrow \boldsymbol{A} + 2\eta_{\boldsymbol{A}}\left(\boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R} + \boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^T\Big). \right. \tag{19}$$

As a final step, in oder to assure the orthogonality constraint on $\boldsymbol{A}$ we project the updated $\boldsymbol{A}$ by considering its QR decomposition [15, 16] $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{T}$, where $\boldsymbol{Q} \in \mathbb{R}^{n\times k}$ is orthogonal and $\boldsymbol{T} \in \mathbb{R}^{k\times k}$ is invertible upper triangular, and following the update $\boldsymbol{A} \leftarrow \boldsymbol{Q}$. This concludes our overview of the methods to come up with appropriate factors. For more detailed information about the alternating least squares solutions for DEDICOM factorization for matrices and tensors we refer the reader to [1, 13, 15, 16].

3 Stochasticity Constrained Nonnegative DEDICOM

In this section we will present a new ALS algorithm to particularly consider DEDICOM factorization with column stochastic loadings and nonnegative affinities for interpretability of the resulting factors. To begin with, as noted in [15,16], we can assure nonnegative affinities $\boldsymbol{R}$ by solving the nonnegative least squares problem introduced in [12] as in (10).

Next considering the theoretical properties of constraining the columns of matrix $\boldsymbol{A}$ to (4), we note that each column resides in the standard simplex Δ^{n-1}, which is the convex hull of the standard basis vectors of $\mathbb{R}^n$. Formally, given that δ_{ij} represents the Kronocker delta and $\boldsymbol{\mathcal{V}} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n | \boldsymbol{v}_i = [\delta_{i1}, \ldots, \delta_{in}]^T\}$ is the set of the standard basis vectors of $\mathbb{R}^n$, the standard simplex Δ^{n-1} is defined as the compact set

$$\Delta^{n-1} = \Big\{ \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \mid \sum_{i=1}^{n} \alpha_i = 1 \wedge \alpha_i \geq 0 \; \forall \; i \; \in [1, \ldots, n] \Big\}. \tag{20}$$

This indicates that, finding an appropriate column stochastic matrix $\boldsymbol{A}$ minimizing the convex objective function in (5) can be reduced down to a constrained optimization problem in the convex compact simplex Δ^{n-1}.

Constrained optimization problems of this kind can be easily tackled using the Frank-Wolfe algorithm [7,9], which is also known as the conditional gradient method [9]. The main idea behind the Frank-Wolfe algorithm is to minimize differentiable convex functions over their domains forming compact convex sets using iterative first-order Taylor approximations until achieving convergence. Formally given a differentiable convex function $f : \boldsymbol{\mathcal{S}} \rightarrow \mathbb{R}$ over a compact convex set $\boldsymbol{\mathcal{S}}$, the Frank-Wolfe algorithm aims to solve

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) \tag{21}$$

for $\boldsymbol{x} \in \boldsymbol{\mathcal{S}}$, by iteratively solving for

$$\boldsymbol{s}_t = \min_{\boldsymbol{s} \, \in \, \boldsymbol{\mathcal{S}}} \; \boldsymbol{s}^T \nabla f(\boldsymbol{x}_t), \tag{22}$$

where $\nabla f(\boldsymbol{x}_t)$ is the gradient of the optimized function f evaluated at the current solution $\boldsymbol{x}_t$. Following that, at each iteration t, the algorithm estimates the new solution by evaluating

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \; \alpha_t(\boldsymbol{s}_t - \boldsymbol{x}_t), \tag{23}$$

where $\alpha_t \in (0, \ldots, 1]$ is the learning rate that is usually selected by performing a line search on (23) or set to be monotonically decreasing as a function of t [7,9]. Additionally, one particular advantage of this optimization method is that it allows for ϵ-approximation for a given maximum number of iterations t_{max} [7,9]. Considering our special case, since the set of vertices of a standard simplex is equivalent to the set of standard basis $\boldsymbol{\mathcal{V}}$ in $\mathbb{R}^n$ [3], at each iteration Frank-Wolfe

algorithm moves the current solution into the direction of one of the standard basis until achieving convergence.

Following that, since the stochasticity constraint does not allow for a simplification as in (17), we require the full derivation of the gradient matrix $\frac{\partial E(\boldsymbol{A})}{\partial \boldsymbol{A}}$ from (13) so as to adapt the Frank-Wolfe algorithm to find appropriate stochastic columns for $\boldsymbol{A}$. To this end we start by considering $E_2(\boldsymbol{A})$ from (16) in terms of a scalar summation as

$$\operatorname{tr}\left[\boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\right] = \sum_u \sum_v \sum_w \sum_x \sum_y \sum_z a_{uv} r_{wv} a_{xw} a_{xy} r_{yz} a_{uz} \tag{24}$$

and consider its partial derivative with respect to an arbitrary element a_{ij} of $\boldsymbol{A}$ that can be written due to linearity of differentiation as

$$\frac{\partial E_2(\boldsymbol{A})}{\partial a_{ij}} = \sum_u \sum_v \sum_w \sum_x \sum_y \sum_z \frac{\partial}{\partial a_{ij}} \left(a_{uv} r_{wv} a_{xw} a_{xy} r_{yz} a_{uz}\right). \tag{25}$$

The expression in (25) can be further simplified by the product rule expansion

$$\begin{aligned}\frac{\partial E_2(\boldsymbol{A})}{\partial a_{ij}} =& \sum_u \sum_v \sum_w \sum_x \sum_y \sum_z \frac{\partial}{\partial a_{ij}} (a_{uv} r_{wv})(a_{xw} a_{xy} r_{yz} a_{uz}) + \\ & \sum_u \sum_v \sum_w \sum_x \sum_y \sum_z \frac{\partial}{\partial a_{ij}} (a_{xw})(a_{uv} r_{wv} a_{xy} r_{yz} a_{uz}) + \\ & \sum_u \sum_v \sum_w \sum_x \sum_y \sum_z \frac{\partial}{\partial a_{ij}} (a_{xy})(a_{uv} r_{wv} a_{xw} r_{yz} a_{uz}) + \\ & \sum_u \sum_v \sum_w \sum_x \sum_y \sum_z \frac{\partial}{\partial a_{ij}} (a_{uz} r_{yz})(a_{uv} r_{wv} a_{xw} a_{xy})\end{aligned} \tag{26}$$

and evaluating the derivatives results in

$$\begin{aligned}\frac{\partial E_2(\boldsymbol{A})}{\partial a_{ij}} =& \sum_w \sum_x \sum_y \sum_z r_{wv} a_{xw} a_{xy} r_{yz} a_{uz} + \\ & \sum_u \sum_v \sum_y \sum_z a_{uv} r_{wv} a_{xy} r_{yz} a_{uz} + \\ & \sum_u \sum_v \sum_w \sum_z a_{uv} r_{wv} a_{xw} r_{yz} a_{uz} + \\ & \sum_v \sum_w \sum_x \sum_y r_{yz} a_{uv} r_{wv} a_{xw} a_{xy}.\end{aligned} \tag{27}$$

Finally, reformulating the four summations in (27) in terms of matrix multiplications as

$$\frac{\partial E_2(\boldsymbol{A})}{\partial a_{ij}} = 2\left[\boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}\right]_{ij} + 2\left[\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}^T\right]_{ij} \tag{28}$$

Randomly initialize valid $\boldsymbol{A}$ and $\boldsymbol{R}$
Let $\boldsymbol{\mathcal{V}} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n | \boldsymbol{v}_i = [\delta_{i1}, \ldots, \delta_{in}]^T\}$
while *Stopping condition is not satisfied* **do**
  //Update $\boldsymbol{A}$ with Frank-Wolfe procedure
  while $t \neq t_{max}$ *and updates of* $\boldsymbol{A}$ *are not small* **do**
    //Define the gradient matrix
    $\boldsymbol{G} = \frac{\partial E(\boldsymbol{A})}{\partial \boldsymbol{A}} = 2\left(\boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R} + \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}^T - \boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R} - \boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^T\right)$
    //Define the monotonically decreasing learning rate
    $\alpha \leftarrow 2/(t+2)$
    for $b \in \{1, \ldots, k\}$ **do**
      //Find the minimizer simplex vertex i
      $i = \underset{l}{\operatorname{argmin}}\, g_{bl}$
      //Update columns $\boldsymbol{a}_b$ of $\boldsymbol{A}$
      $\boldsymbol{a}_b \leftarrow \boldsymbol{a}_b + \alpha(\boldsymbol{v}_i - \boldsymbol{a}_b)$
    //Update the iterator
    $t \leftarrow t + 1$
  //Update $\boldsymbol{R}$ by solving the nonnegative least squares problem
  $\boldsymbol{R} \leftarrow \underset{\boldsymbol{R}}{\operatorname{argmin}} \left\|\operatorname{vec}(\boldsymbol{S}) - \left(\boldsymbol{A} \otimes \boldsymbol{A}\right)\operatorname{vec}(\boldsymbol{R}))\right\|^2$ w.r.t $r_{ij} \geq 0 \; \forall \, i, j$

Alg. 1: A Frank-Wolfe based *projection-free* ALS algorithm to find Semi-nonnogative DEDICOM factors with column stochastic loading matrix $\boldsymbol{A}$ and nonnegative affinity matrix $\boldsymbol{R}$. At each iteration of the algorithm, we alternate between finding an optimal column stochastic $\boldsymbol{A}$ keeping $\boldsymbol{R}$ fixed and finding a nonnegative matrix $\boldsymbol{R}$ keeping $\boldsymbol{A}$ fixed. Note that the learning rate here is set to decrease monotonically.

allows us to define the gradient matrix $\frac{\partial E_2(\boldsymbol{A})}{\partial \boldsymbol{A}}$ as

$$\frac{\partial E_2(\boldsymbol{A})}{\partial \boldsymbol{A}} = 2\Big(\boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R} + \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}^T\Big). \tag{29}$$

Combining the results from (18) and (29) the full gradient matrix of the minimized error norm for $\boldsymbol{A}$ is calculated as follows

$$\frac{\partial E(\boldsymbol{A})}{\partial \boldsymbol{A}} = 2\left(\boldsymbol{A}\boldsymbol{R}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R} + \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}^T - \boldsymbol{S}^T\boldsymbol{A}\boldsymbol{R} - \boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^T\right), \tag{30}$$

which allows us to define a column-wise Frank-Wolfe algorithm to come up with optimal column stochastic loading matrix $\boldsymbol{A}$ minimizing (5). We list the essential steps of our adaptation of the projection free Frank-Wolfe algorithm in Alg. 1. In a nutshell, our algorithm starts by randomly initializing (valid) matrices $\boldsymbol{A}$ and $\boldsymbol{R}$, continues with alternating between the Frank-Wolfe optimization updates to come up with optimal column stochastic $\boldsymbol{A}$ and alternating least squares

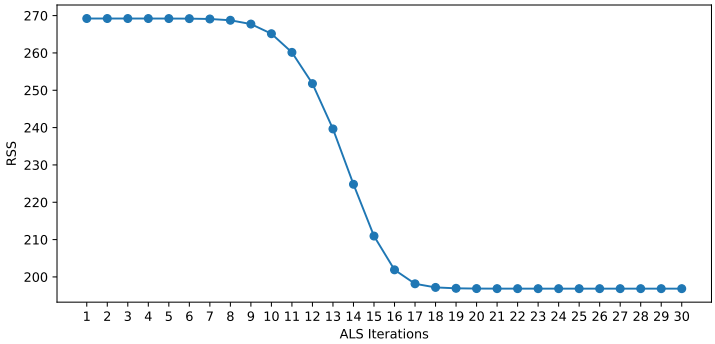

Fig. 2: Illustration of the evolution of the Residual Sum of Squares (RSS) for factorizing our empirical conditional probability matrix indicating asymmetric relations between games using DEDICOM with $k = 3$. Our provably converging algorithm monotonically decreases the RSS at each iteration.

updates for nonnegative $\boldsymbol{R}$ until a predefined stopping condition is met. The stopping conditions are usually selected based on reaching a maximum number of alternating least squares updates (not to be mixed with the maximum iteration count t_{max} of the Frank-Wolfe updates for columns of $\boldsymbol{A}$ in Alg. 1), having minor subsequent changes in the minimized objective (for our case the matrix norm in (5)) or combining both of these conditions.

4 A Business Intelligence Application: Analyzing Asymmetric Game Ownership Associations in an Online Gaming Platform

In order to evaluate our algorithm, we consider a use-case from game analytics [14, 16], where we analyze the asymmetric relationships among various types of games. To this end we consider a dataset from an online gaming platform called Steam, which hosts thousands of games of various genres to a large player-base with its size ranging from 9 to 15 million[3] (daily active) players. We used the dataset from [14] which contains game ownership information of more than six million users about 3007 titles.

Representing the ownership information as a *bipartite* matrix $\boldsymbol{Y} \in \{0,1\}^{m \times n}$ indicating ownership of information of m players for n games, we construct the asymmetric association among games in terms of their pairwise empirical conditional probabilities. To this end, we define the directional similarity from game i to game j as

$$s_{ij} = \frac{|\{c \,|y_{ci} \neq 0 \,\forall\, c\} \cap \{b \,|y_{bj} \neq 0 \,\forall\, b\}|}{|\{c \,|y_{ci} \neq 0 \,\forall\, c\}|}, \tag{31}$$

[3] `http://store.steampowered.com/stats/`

Loadings	Identifier	Dominant Games
$\boldsymbol{a}_1$	*TF 2/FPS*	TF 2, CS: Source and Red Orchestra: Ostfront 41-45
$\boldsymbol{a}_2$	*Indie/Platformer*	WSS, Ethan and Oozi: Earth Adventure
$\boldsymbol{a}_3$	*Flagships/AAA*	Left 4 Dead 2, Dota 2, Skyrim, HL 2, Garry's Mod

Table 1: Selection of games with high loading values for the analyzed dataset.

where $|\cdot|$ indicates set cardinality and y_{pq} represents if the pth player owns the qth game. It is important to note that (31) is inherently asymmetric and a special case of the so-called Tversky index [18] with *prototype* and *variant* weights of respectively 1 and 0 (or vice versa). Since the number of computations required to obtain the similarity matrix $\boldsymbol{S}$ scales to $O(mn^2)$, we parallelized the procedure of computing the similarity values in (31) by utilizing a hybrid parallelization technique that simultaneously exploits both distributed and shared memory architectures in order to speedup the computation time.

Having obtained the asymmetric similarity matrix with the empirical conditional probability values in $\boldsymbol{S}$, we factorize it using our algorithm and present the results with $k = 3$. To explicitly ignore modeling the self-loops, at each alternating least squares iteration, we replaced the diagonal of $\boldsymbol{S}$ with the diagonal of its current reconstruction [1]. In Fig. 2 we show the residual sum of squares, which decreases monotonically and converges to a minimum after 30 iterations. Analyzing the resulting factors, we observe that the resulting two columns (or modes) $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ of $\boldsymbol{A}$ are sparse whereas the last column was dense in terms of non-zero probability values, where the most dominant games vary from column to column. In Tbl. 1 we present the games with high loadings in their corresponding column and observe that mostly the free-to-play flagship game Team Fortress 2 (TF 2), followed by the FPS games CS: Source and Red Orchestra: Ostfront 41-45, contribute to $\boldsymbol{a}_1$.On the other hand, the indie-platformer games Wooden Sen'SeY (WSS), Ethan (Meteor Hunter) and Oozi: Earth Adventure contribute mostly to $\boldsymbol{a}_2$. Finally, the mostly dense $\boldsymbol{a}_3$ has very high loadings for all of the flagship games of the analyzed platform and other AAA games including Left 4 Dead 2, Dota 2, Skyrim, Half-life 2 (HL 2) and Garry's Mod.

Analyzing the resulting row-normalized affinity matrix that encodes the asymmetric relations between the modes from Fig. 3, the first rows indicates tendencies of TF 2 players more to the indie-platformer mode than to the AAA games because of the fact that TF 2 is mostly a singular game that people primarily prefer in the Steam platform [14]. On the other hand, inline with the results from [14] the opposite directional associations, namely, from the flagships to the TF 2 are high which is related to the fact that majority of the players playing one or more of the flagship games also prefer TF 2 [14]. Finally analyzing the second mode, similar to the results in [15] the self association is the highest association we observe (indicating players preferring mostly to stay in the same genre), this is followed by high associations to the TF 2 and the mode related to the flagship and AAA games.

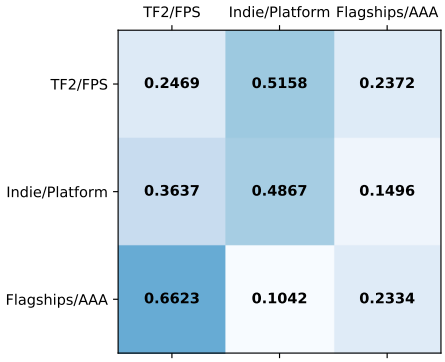

Fig. 3: Results of factorizing the empirical conditional probability graph created for the pair-wise game ownership information of more than six million players of an online gaming platform. The normalized nonnegative affinity matrix $\boldsymbol{R}$ shows relationships between groups automatically determined by the DEDICOM algorithm.

It is worth mentioning that, parallel to the results in [1], by increasing the number of modes k for the loading matrix $\boldsymbol{A}$, we observed more detailed partitions regarding the modeled patterns. Running our algorithm with $k = 2$, we observed a higher reconstruction error and that the indie-platformer mode $\boldsymbol{a}_2$ remains to be one of the factors whereas the games contributing to $\boldsymbol{a}_3$ have merged with ones contributing to mode $\boldsymbol{a}_1$. On the other hand, considering the resulting factors when we ran our algorithm with $k = 4$, we obtained a slight improvement regarding the reconstruction error and observed that the modes $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ remained the same, however, mode $\boldsymbol{a}_3$ has split into two different modes where both contain the same games as in $\boldsymbol{a}_3$ and the new mode contains additional indie and plaformer games (reducing the probability on the games of $\boldsymbol{a}_3$) such as Finding Teddy, Gentlemen, Gravi and Face Noir.

5 Conclusion and Future Work

In this work we studied the DEDICOM model to factorize asymmetric similarity matrices into a combination of a low rank loading matrix and an affinity matrix. We gave an overall overview about the theoretical details and algorithms to come up with appropriate factors. In essence, the affinity matrix matrix $\boldsymbol{R}$ indicates different directional structures that are determined by the columns of the loading matrix $\boldsymbol{A}$. Having defined our objective function to be the matrix norm of the difference between the factorized asymmetric similarity matrix and its DEDICOM reconstruction, we derived the gradient matrix for this function with respect to the loading matrix $\boldsymbol{A}$ and presented a variant of the Frank-Wolfe algorithm to obtain interpretable column stochastic loadings and nonnegative affinities.Running our algorithm on a dataset containing asymmetric pairwise

relationships between more than 3000 games, we found interesting patterns indicating directional preferences among games.

Our future work involves analyzing the performance of our model for different tasks including link prediction and representation learning [13, 16]. Considering the tensor extensions in [13,16], our future work also involves extending the proposed model to tensors factorizations which can allow us to analyze asymmetric similarity matrices from different sources. In the context of business intelligence and game analytics, this might allow us to analyze and compare, for instance, preferences of different countries or player groups.

References

1. Bader, B., Harshman, R., Kolda, T.: Temporal Analysis of Semantic Graphs using ASALSAN. In: Proc. of IEEE ICDM (2007)
2. Bauckhage, C., Sifa, R., Drachen, A., Thurau, C., Hadiji, F.: Beyond Heatmaps: Spatio-Temporal Clustering using Behavior-Based Partitioning of Game Levels. In: Proc. of IEEE CIG (2014)
3. Bauckhage, C.: k-Means Clustering via the Frank-Wolfe Algorithm. In: Proc. of KDML-LWDA (2016)
4. Chew, P.A., Bader, B.W., Rozovskaya, A.: Using DEDICOM for Completely Unsupervised Part-Of-Speech Tagging. In: Proc. Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics (2009)
5. Cremonesi, P., Koren, Y., Turrin, R.: Performance of Recommender Algorithms on Top-N Recommendation Tasks. In: Proc. ACM Recsys (2010)
6. Cutler, A., Breiman, L.: Archetypal Analysis. Technometrics 36(4), 338–347 (1994)
7. Frank, M., Wolfe, P.: An Algorithm for Quadratic Programming. Naval Research Logistics 3(1-2) (1956)
8. Harshman, R.A.: Models for Analysis of Asymmetrical Relationships among N Objects or Stimuli. In: Proc. Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology (1978)
9. Jaggi, M.: Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. In: Proc. of ICML (2013)
10. Kantor, P., Rokach, L., Ricci, F., Shapira, B.: Recommender Systems Handbook. Springer (2011)
11. Koren, Y., Bell, R., Volinsky, C.: Matrix Factorization Techniques for Recommender Systems. Computer 42(8) (2009)
12. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems. SIAM (1974)
13. Nickel, M., Tresp, V., Kriegel, H.: A Three-way Model for Collective Learning on Multi-relational Data. In: Proc. of ICML (2011)
14. Sifa, R., Drachen, A., Bauckhage, C.: Large-Scale Cross-Game Player Behavior Analysis on Steam. In: Proc. of AAAI AIIDE (2015)
15. Sifa, R., Ojeda, C., Bauckhage, C.: User Churn Migration Analysis with DEDICOM. In: Proc. of ACM RecSys (2015)
16. Sifa, R., Srikanth, S., Drachen, A., Ojeda, C., Bauckhage, C.: Predicting Retention in Sandbox Games with Tensor Factorization-based Representation Learning. In: Proc. of IEEE CIG (2016)
17. Sifa, R., Bauckhage, C., Drachen, A.: Archetypal Game Recommender Systems. In: Proc. of KDML-LWA (2014)
18. Tversky, A.: Features of Similarity. Psychological Review 84(4) (1977)