Data Mining and Knowledge Discovery (2024) 38:2306–2347
https://doi.org/10.1007/s10618-024-01033-y

The Hadamard decomposition problem

Martino Ciaperoni[1] · **Aristides Gionis**[2] · **Heikki Mannila**[1]

Received: 5 December 2023 / Accepted: 28 April 2024 / Published online: 21 May 2024

Abstract

We introduce the *Hadamard decomposition problem* in the context of data analysis. The problem is to represent exactly or approximately a given matrix as the Hadamard (or element-wise) product of two or more low-rank matrices. The motivation for this problem comes from situations where the input matrix has a multiplicative structure. The Hadamard decomposition has potential for giving more succint but equally accurate representations of matrices when compared with the gold-standard of singular value decomposition (SVD). Namely, the Hadamard product of two rank-h matrices can have rank as high as h^2. We study the computational properties of the Hadamard decomposition problem and give gradient-based algorithms for solving it approximately. We also introduce a mixed model that combines SVD and Hadamard decomposition. We present extensive empirical results comparing the approximation accuracy of the Hadamard decomposition with that of the SVD using the same number of basis vectors. The results demonstrate that the Hadamard decomposition is competitive with the SVD and, for some datasets, it yields a clearly higher approximation accuracy, indicating the presence of multiplicative structure in the data.

Keywords Matrix decomposition · Hadamard product

Responsible editor: Panagiotis Papapetrou.

✉ Martino Ciaperoni
martino.ciaperoni@aalto.fi

Aristides Gionis
argioni@kth.se

Heikki Mannila
heikki.mannila@aalto.fi

[1] Department of Computer Science, Aalto University, Helsinki, Finland

[2] Division of Theoretical Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

1 Introduction

Matrix decomposition (or matrix factorization) methods aim to obtain simpler or more succinct exact or approximate representations of an input matrix.[1] There are many different decomposition methods, and most of them aim at representing an input matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$ as the matrix product of two or more matrices which are in some sense simpler than the matrix $\mathbf{D}$.

For example, the singular value decomposition (SVD), a frequently-used decomposition method, represents the input matrix as $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, a product of an orthonormal matrix, a diagonal matrix, and (the transpose of) another orthonormal matrix.[2] There are many other decomposition methods of the same type (Bernstein 2018; Bro and Smilde 2014; Gillis 2020).

An alternative interpretation of SVD is that it provides a way of representing the matrix $\mathbf{D}$ as the *sum* of simpler building-block matrices. For SVD, the simpler building blocks are rank-1 matrices, i.e., matrices defined as the outer product of row and column vectors. If $\mathbf{D}$ has rank r, then the representation of $\mathbf{D}$ as a sum of rank-1 matrices has r terms. If the $k < r$ terms corresponding to the largest diagonal values in $\mathbf{\Sigma}$ are used to represent $\mathbf{D}$, a rank-k approximate representation of $\mathbf{D}$ is obtained, which in fact is the optimal rank-k approximation of $\mathbf{D}$ with respect to Frobenius norm (Golub and Van Loan 2013).

In this paper, we consider a more complex type of matrix decomposition, which relies on the *Hadamard matrix product* to introduce a multiplicative layer. The Hadamard (or element-wise) matrix product $\mathbf{D}_1 \odot \mathbf{D}_2$ of two matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ of the same dimensions has a very simple definition: each entry in $\mathbf{D}_1 \odot \mathbf{D}_2$ is the product of the corresponding entries in $\mathbf{D}_1$ and $\mathbf{D}_2$.

The main question considered in this paper is the following.

1.1 Problem (informal)

Given a matrix $\mathbf{D}$, find a good representation of $\mathbf{D}$ as the Hadamard product $\mathbf{D}_1 \odot \mathbf{D}_2$ of two low-rank matrices $\mathbf{D}_1$ and $\mathbf{D}_2$.

The decomposition $\mathbf{D} = \mathbf{D}_1 \odot \mathbf{D}_2$ is referred to as Hadamard decomposition.

Note that taking logarithms, when possible, would change the Hadamard product to an element-wise sum, but it would also destroy the low-rank property of the arguments $\mathbf{D}_1$ and $\mathbf{D}_2$, as $\log(\mathbf{D}_1)$ can have an arbitrarily high rank even if $\mathbf{D}_1$ has low rank.

1.2 Intuition behind the Hadamard decomposition model

We next describe the basic intuition behind the proposed model: why are we interested in Hadamard decompositions and decompositions to low-rank matrices?

[1] We work with real-valued matrices, unless otherwise indicated.

[2] In the SVD representation $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ we can assume that the matrices $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{V}$ have dimensions $n \times r$, $r \times r$, and $r \times m$, respectively, for an input matrix $\mathbf{D}$ of dimensions $n \times m$.

First, why Hadamard decompositions? The answer is that many natural phenomena are multiplicative or conjunctive in nature. In medicine, the theme of additive and multiplicative risks has been discussed quite intensively (Breslow et al. 1983; Qi et al. 2009; Lee 2001). Similarly, in genetics, genotypes and environment may interact multiplicatively (Roberto Cruz 1992).

For a more detailed example, in ecology and paleontology, the suitability of a location for a particular species typically depends on several factors (Thompson 1999; Thompson et al. 2018). To give a very simplified example, a species can live in a certain area if the temperature is high enough and if there is enough water available. Note that this condition is conjunctive in nature, and thus, corresponds to multiplication of 0–1 variables.

More concretely, consider a paleontological dataset (Fortelius et al. 2006; Fortelius 2008) where the rows correspond to locations in which fossils have been found, and the columns correspond to taxa (in this case, genera of land mammals). The data is binary, with the 1's corresponding to occurrences of the genus in that location. The data is not perfect, and especially a 0-entry is not a strong indication of the nonexistence of the genus.

For such a dataset it is quite natural to search for a decomposition of the occurrences of genera, i.e., for the entries equal to 1, by a multiplicative model. The fascinating paleontological question is then to search for an ecological interpretation of the factors in the multiplicative model. The study of the ecological interpretation of the retrieved multiplicative factors is beyond the scope of this paper.

Given the prevalence of multiplicative phenomena in the real world, it is not surprising that multiplicative models are widely employed across various domains of data analysis, including regression, time series and network analysis (Cox 1984; Ursu and Duchesne 2009; Hoff 2021). Nonetheless, prior to this work, the potential for data analysis of a matrix decomposition model based on element-wise multiplication has not been thoroughly explored.

Finally, one can ask the question of why restricting the Hadamard decomposition to low-rank matrices? The first answer is that low rank is an indication of simplicity: low-rank matrices are easier to interpret than general matrices. Also, the insightful paper of Udell and Townsend (Udell and Townsend 2019) argues that datasets stemming from simple generative models have low rank.

1.3 Hadamard decomposition and singular value decomposition

Assume that the matrix $\mathbf{D}$ has dimensions $n \times m$ and the matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ (with the same dimensions) have rank h.[3] Then, we can write $\mathbf{D}_1 = \mathbf{A}_1\mathbf{B}_1$ and $\mathbf{D}_2 = \mathbf{A}_2\mathbf{B}_2$ for matrices $\mathbf{A}_i$ with dimensions $n \times h$ and $\mathbf{B}_i$ with dimensions $h \times m$, for $i = 1, 2$.

The Hadamard decomposition of $\mathbf{D}$ into the rank-h matrices $\mathbf{D}_1$ and $\mathbf{D}_2$, if it exists, can be written as a Hadamard product of two sums of inner products:

[3] We always assume an input matrix $\mathbf{D}$ of dimensions $n \times m$, unless otherwise indicated.

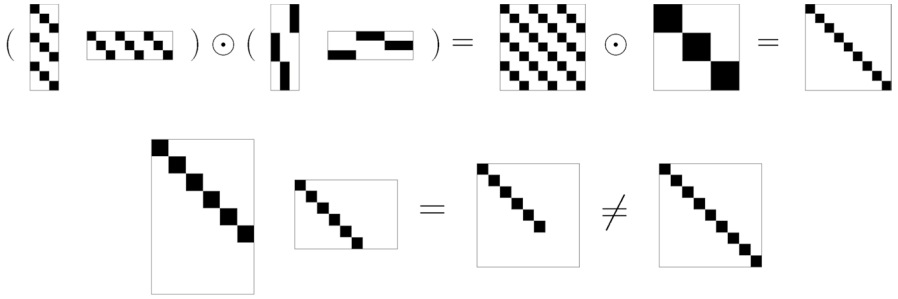

Fig. 1 Example: Hadamard decomposition and SVD of the identity matrix. Top: the identity matrix represented as a Hadamard product of two rank-3 matrices. Bottom: the identity matrix represented via rank-6 SVD. Black squares represent 1-entries. The Hadamard decomposition uses two sets of 6 basis vectors to reconstruct the identity matrix with no error, whereas the SVD with the same number of basis vectors cannot reconstruct the identity matrix exactly

$$\mathbf{D}_{i,j} = \mathbf{D}_{1_{i,j}} \mathbf{D}_{2_{i,j}} = \left(\sum_{u=1}^{h} \mathbf{A}_{1_{i,u}} \mathbf{B}_{1_{u,j}} \right) \left(\sum_{u=1}^{h} \mathbf{A}_{2_{i,u}} \mathbf{B}_{2_{u,j}} \right). \tag{1}$$

Since $\mathbf{D}_1$ and $\mathbf{D}_2$ have rank h, their Hadamard product has rank at most h^2, as can be easily shown from Eq. (1) by using the distributive law. Namely, we have:

$$\mathbf{D}_{i,j} = \sum_{u=1}^{h} \sum_{v=1}^{h} \mathbf{A}_{1_{i,u}} \mathbf{B}_{1_{u,j}} \mathbf{A}_{2_{i,v}} \mathbf{B}_{2_{v,j}} = \sum_{u=1}^{h} \sum_{v=1}^{h} (\mathbf{A}_{1_{i,u}} \mathbf{A}_{2_{i,v}})(\mathbf{B}_{1_{u,j}} \mathbf{B}_{2_{v,j}}), \tag{2}$$

and this is a sum of h^2 terms. Furthermore, this bound is tight, as can be seen, e.g., from the example in Fig. 1, which shows how to represent the rank-9 identity matrix as the Hadamard product of two rank-3 matrices.

The above observation implies that there are cases where the Hadamard decomposition is more efficient than SVD: using two rank-h matrices, the Hadamard decomposition can represent matrices of rank h^2, while the SVD with the same number of vectors (or parameters) can represent exactly only matrices up to rank $2h$.

An example of the advantages given by the Hadamard decomposition over the SVD is provided in Fig. 1. We consider the expressive power of the Hadamard decomposition by looking at identity matrices. Figure 1 (*top*) shows how we can represent the 9×9 identity matrix as the Hadamard decomposition of two rank-3 matrices, i.e., with 6 vectors of length 9. If instead we use the SVD with the same number of vectors, we get the result shown in Fig. 1 (*bottom*), suggesting that we cannot represent the 9×9 identity matrix.

We turn to real data. For instance, for the task of image compression, the Hadamard decomposition leads to a more accurate reconstruction of an input image, compared to the SVD using the same number of parameters for reconstruction. Such an example for image compression is shown in Fig. 2.

Fig. 2 Example: Hadamard decomposition and SVD in the task of image compression. Both the Hadamard decomposition and the SVD use 20 basis vectors for compression. Although the Hadamard decomposition and the SVD rely on the same total number of vectors to represent the input image, the reconstruction provided by the Hadamard decomposition is of higher quality. In particular, the Hadamard decomposition incurs a total sum of squared errors lower than the SVD by 43%

While the Hadamard decomposition can be more efficient than the SVD, the SVD can represent all matrices of rank $2h$ as the sum of two rank-h matrices. On the other hand, an arbitrary matrix of rank h^2 matrix does not necessarily have a representation as the Hadamard product of two rank-h matrices. This can be seen from a simple dimensionality argument. Consider a matrix $\mathbf{D}$ of dimensions $h^2 \times h^2$ and rank h^2. The Hadamard decomposition $\mathbf{D} = \mathbf{D}_1 \odot \mathbf{D}_2$, where $\mathbf{D}_1$ and $\mathbf{D}_2$ are rank-h factors, encodes a system of nonlinear equations where the entries of $\mathbf{D}$ are viewed as constants and the entries of $\mathbf{D}_1$ and $\mathbf{D}_2$ are viewed as variables. This system includes $h^2h^2 = h^4$ equations (one per entry of $\mathbf{D}$) and, because of the low-rank constraint on the two Hadamard factors $\mathbf{D}_1$ and $\mathbf{D}_2$, only $(h^2h + hh^2) = 2h^3$ variables. Thus, for $h > 2$, there are more equations than variables. Intuitively, this implies that all the equations will be simultaneously satisfied only in special cases. For instance, one can show that if the matrix $\mathbf{D}$ includes a row or a column with all but a single entry being 0, then not all the equations in the system under consideration can be satisfied.

As a consequence, we do not pursue the goal of retrieving an exact Hadamard decomposition. Rather, we seek to find a Hadamard decomposition that offers a good approximation of the input matrix.

1.4 (Non)uniqueness

It is obvious that the above problem does not have a unique answer: if $\mathbf{D}_1 \odot \mathbf{D}_2$ is a good representation of $\mathbf{D}$, then for any nonzero scalar c also $(c\mathbf{D}_1) \odot (1/c)\mathbf{D}_2$ is another representation of $\mathbf{D}$ with the same accuracy. There are other sources of nonuniqueness in the Hadamard decomposition. While some of them are trivial (e.g., reordering the factors), as we show in Sect. 6, even for simple matrices, there can be several clearly different Hadamard decompositions.

1.5 Algorithms

The algorithmic problem of finding a Hadamard decomposition is easy to formulate as a multivariate optimization problem: given $\mathbf{D}$, find matrices $\mathbf{A}_1$, $\mathbf{B}_1$, $\mathbf{A}_2$ and

$\mathbf{B}_2$ minimizing some norm of $\mathbf{D} - ((\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2))$. However, finding a combinatorial algorithm for this problem is challenging. Instead, we use gradient-descent methods (Ruder 2016), which are effective for our purposes. More specifically, we devise alternating gradient-descent algorithms that, in practice, exhibit good convergence properties and find a Hadamard decomposition which can outperform the SVD when the goal is to represent the input matrix $\mathbf{D}$ as concisely as possible.

1.6 Contributions

The contributions of this paper are as follows.

- We introduce a novel matrix decomposition problem based on the Hadamard product of low-rank factors.
- We analyze the Hadamard decomposition model, its relationship with the singular value decomposition, and its uniqueness properties.
- We also investigate a mixed Hadamard decomposition model that combines the SVD and the Hadamard decomposition.
- We give simple gradient-descent algorithms for computing the Hadamard decomposition and the mixed Hadamard decomposition.
- We describe extensive experiments in synthetic and real-world data to investigate the performance of the Hadamard decomposition.

1.7 Structure of the paper

The rest of this paper is organized as follows. In Sect. 2 we describe the related work, and in Sect. 3 we introduce the necessary notation and some basic concepts. In Sect. 4 we present the Hadamard decomposition model and give a more formal definition of the Hadamard decomposition problem. Section 5 presents the mixed Hadamard decomposition model which combines Hadamard decomposition and SVD. The issue of (non)uniqueness of the Hadamard decomposition is considered in Sect. 6, while Sect. 7 describes the algorithms for computing the (mixed) Hadamard decomposition. Section 8 presents our thorough experimental evaluation, and finally, Sect. 9 presents our conclusions.

2 Related work

There is an extensive body of literature on matrix decomposition techniques.

2.1 Singular value decomposition

As discussed in Sect. 1, the *singular value decomposition* (SVD) (Klema and Laub 1980) is a very widely used matrix decomposition method. The rank-k SVD enjoys a desirable property that plays an important role in numerous applications: it provides the best possible approximation of $\mathbf{D}$ among all matrices of rank k in terms of (squared) Frobenius and spectral norm of the approximation error (the Eckart-Young-Mirsky theorem (Stewart 1993)). The SVD, which is formally defined in Sect. 3, is used as a point of comparison when investigating the approximation accuracy of the Hadamard decompositon.

2.2 Other matrix decomposition techniques

Besides SVD, any $n \times m$ matrix $\mathbf{D}$ of rank r admits other decompositions of the form $\mathbf{XY}$, with $\mathbf{X} \in \mathbb{R}^{n \times r}$ and $\mathbf{Y} \in \mathbb{R}^{r \times m}$ (Banerjee and Roy 2014). There are several related decomposition methods posing different conditions for the matrices $\mathbf{X}$ and $\mathbf{Y}$. Examples include the QR, Cholesky, and LU decompositions; see, e.g., Bernstein (2018) for many additional references.

The SVD is supported by good theoretical properties and finds application in many different contexts. However, it has its drawbacks: for instance, given a non-negative input matrix, the SVD may yield negative values, which can be difficult to interpret. To address this drawback, methods for *non-negative matrix factorization* (NMF) (Gillis 2020) have been widely studied. Similarly, Boolean matrix decomposition looks for factors with Boolean values and using the Boolean operators '`or`' and '`and`' (see, e.g., DeSantis et al. 2022; Miettinen et al. 2008; Miettinen and Vreeken 2014). There have also been studies in matrix decomposition from the probabilistic point of view (Mnih and Salakhutdinov 2007; Salakhutdinov and Mnih 2008). More recently, deep matrix factorization has attracted attention (De Handschutter et al. 2021). In deep matrix factorization, the input matrix is decomposed into the product of multiple factors.

2.3 Non-standard matrix products

In addition to the standard matrix product, alternative product operators for matrices have been studied. Van Loan and Pitsianis (1993) investigate the problem of approximating an input matrix as the Kronecker product of two matrices. While the Hadamard product and the Kronecker product are related, the problem studied by Van Loan and Pitsianis (1993) admits a closed-form solution in terms of the SVD of the input matrix, whereas the problem we study does not admit a similar closed-form solution.

2.4 Hadamard product and decompositions

While the Hadamard product is a non-conventional operator in linear algebra, its fundamental properties have been thoroughly investigated (Horn 1990; Visick 2000; Horn and Yang 2020).

Furthermore, several recent works leverage the Hadamard product. In the field of deep learning, it has been used to improve performance in a computer-vision task (Kim et al. 2016) and to improve convolutional neural networks (Wu 2018).

The Hadamard product is also relevant in multivariate statistics (Styan 1973), for example for probabilistic models such as restricted Boltzmann machines (Montúfar 2018) and in the context of regression, where Hoff (2017) considers parametrizations based on the Hadamard product, and shows that such parametrizations can be used to express L_q penalties, with $q \leq 1$, as sums of L_2 penalties.

In signal processing, Yang et al. (2019) introduce a smoothed covariance matrix defined as a Hadamard product of two matrices.

To address the task of image dehazing, Liu et al. (2020) develop a novel method which is based on the Hadamard product of matrices of pixels.

In 2021, a factorization analogous to the one considered in our work has been explored in connection with federated learning (Hyeon-Woo et al. 2021), where it is used as a reparametrization for neural network architectures with the aim of reducing the communication cost in federated learning. The similarities with our work are limited to the basic expression of the Hadamard-product-based decomposition.

In the algebraic geometry literature, Friedenberg et al. (2017) study Hadamard products of algebraic varieties and introduce a notion of Hadamard decomposition that is equivalent to the definition of Hadamard decomposition adopted in our work. The similarities between our paper and the work of Fiedenberg et al. are limited to the basic formulation of the Hadamard decomposition model. Further, while we focus on the decomposition with two Hadamard factors, they focus on the decomposition with any number of Hadamard factors and they prove insightful bounds on the minimum number of Hadamard factors needed to reconstruct all $n \times m$ matrices using Hadamard factors of rank up to h.

Finally, in 2023, Oneto and Vannieuwenhoven (2023) study, from the point of view of algebraic geometry, the Hadamard-Hitchcock decomposition, which decomposes a tensor as a Hadamard product of several tensor rank decompositions. They establish generic identifiability for the Hadamard-Hitchcock decomposition and propose an algorithm to compute it. The identifiability of the Hadamard-Hitchcock decomposition hinges upon classic results on the identifiability of tensor rank decompositions, which do not transfer to the setting studied in this work. Nonetheless, the techniques and methods of Oneto and Vannieuwenhoven (2023) might be useful in addressing one of our main questions left open: understanding the forms of nonuniqueness in Hadamard decompositions.

2.5 Alternating gradient descent for matrix decomposition

The algorithms we propose to compute Hadamard decompositions are based on alternating gradient descent. Such technique has been widely used for matrix decomposition and recently convergence of alternating gradient descent for *standard* matrix decompositions $\mathbf{D} = \mathbf{XY}$ has been studied (Li and Liang 2017; Ye and Du 2021). To improve the performance of the proposed gradient-descent algorithms, we also leverage a scaling factor proposed in recent work (Tong et al. 2021).

3 Preliminaries

Before we introduce the main ideas behind the proposed methods, we introduce the notation and preliminary notions that are used throughout the paper.

3.1 Notation and basic definitions

Matrices are denoted by upper-case boldface letters, e.g., $\mathbf{D}$. The Frobenius norm of a matrix is denoted by $\|\mathbf{D}\|_F$. The i-th row and j-th column of a matrix $\mathbf{D}$ are denoted by $\mathbf{D}_{i,:}$ and $\mathbf{D}_{:,j}$, respectively. Other vectors are denoted by lower-case boldface letters, e.g., $\mathbf{u}$. Finally, sets are denoted by upper-case roman letters, and scalars by lower-case roman letters.

Given two matrices $\mathbf{A}$ and $\mathbf{B}$ of dimensions $n \times p$ and $p \times m$, respectively, their standard matrix product is a matrix $\mathbf{P} = \mathbf{AB}$ of dimensions $n \times m$ where $\mathbf{P}_{i,j}$ is obtained as the dot product between the i-th row of $\mathbf{A}$ and j-th column of $\mathbf{B}$.

Furthermore, given two matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ both of dimensions $n \times m$, we define their Hadamard (or element-wise) product as the matrix $\mathbf{D} = \mathbf{D}_1 \odot \mathbf{D}_2$ of dimensions $n \times m$, where entry $\mathbf{D}_{i,j}$ is obtained as the (scalar) product $[\mathbf{D}_1]_{i,j}[\mathbf{D}_2]_{i,j}$.

We next state the simple bounds on the rank of a Hadamard product and on expressing a low-rank matrix as a Hadamard product.

Proposition 1 (See, e.g., Styan 1973; Horn 1990.) *If $\mathbf{D}_1$ and $\mathbf{D}_2$ are rank-h matrices, then their Hadamard product $\mathbf{D}_1 \odot \mathbf{D}_2$ has rank at most h^2. On the other hand, if $\mathbf{D}$ is a rank-h matrix, then $\mathbf{D}$ can be represented as a Hadamard product of a rank-h matrix and a rank-1 matrix.*

The first part follows immediately from Eq. 2. The second part follows since any $\mathbf{D}$ can be represented as a Hadamard product $\mathbf{D} \odot \mathbf{1}$, where $\mathbf{1}$ is the identity matrix with respect to the Hadamard product, i.e., the matrix whose entries are all 1.

3.2 Singular value decomposition

The SVD decomposes a matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$ of rank $r \leq \min\{m, n\}$ into three matrices: $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Here, $\mathbf{U}$ is an orthonormal matrix whose columns $\mathbf{u}_i$ are the so-called left singular vectors, $\mathbf{\Sigma}$ is a diagonal matrix whose diagonal entries are the singular values σ_i (conventionally assumed to be in non-increasing order) and $\mathbf{V}$ is an orthonormal matrix whose columns $\mathbf{v}_i$ are the so-called right singular vectors. Alternatively, the SVD can be expressed as a sum of rank-1 matrices, as follows:

$$\mathbf{D} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T. \tag{3}$$

The truncated SVD of rank $k < r$ (or simply rank-k SVD) is obtained by setting to 0 all but the highest k singular values (i.e., diagonal entries) in $\mathbf{\Sigma}$. Hence, the truncated

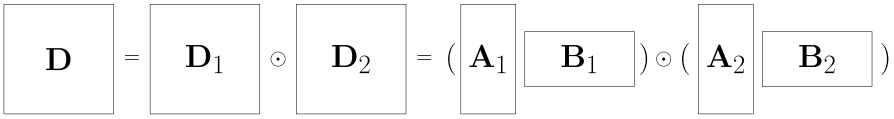

Fig. 3 Diagram illustrating the Hadamard decomposition

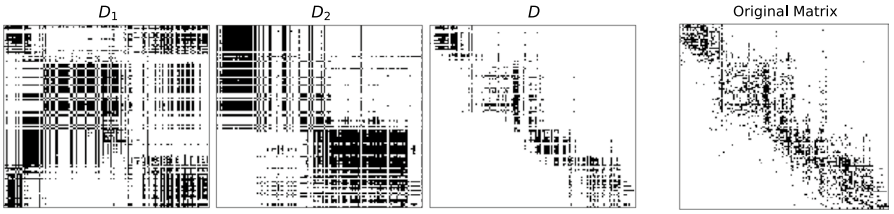

Fig. 4 Example: Hadamard decomposition in the boolean PALEO dataset (Fortelius 2008). Black squares represent 1 entries. We show the (estimated) Hadamard factors $\mathbf{D}_1$ and $\mathbf{D}_2$ of rank 3 and the Hadamard product $\mathbf{D}$, as well as the original dataset approximated with the Hadamard product of two low-rank factors. The entries of the three matrices are rounded to 0 and 1 using 0.5 as a cutoff

SVD of rank k represents $\mathbf{D}$ concisely using two sets of k singular vectors (scaled by singular values), and provides a low-rank approximation of the matrix $\mathbf{D}$.

3.3 Comparing Hadamard decomposition and SVD: the number of parameters

The number of parameters required by a Hadamard decomposition with two rank-h factors is $2(n+m)h$. Similarly, the rank-k SVD is specified by $(n+m)k$ parameters, assuming that $\mathbf{\Sigma}$ is "absorbed" into $\mathbf{U}$ and $\mathbf{V}$. Therefore, it is appropriate to compare the rank-$2h$ SVD and the Hadamard decomposition with two rank-h factors, which use the same number of parameters. Both the rank-$2h$ SVD and the Hadamard decomposition with two rank-h factors include $2h$ length-n basis vectors of the column space of $\mathbf{D}$ and $2h$ length-m basis vectors of the row space of $\mathbf{D}$. Thus, throughout the paper, rather than counting the number of parameters, we measure the size of the representations by using the number of basis vectors for either the row or column space.

4 The Hadamard decomposition problem

In this section, we introduce the problem of Hadamard decomposition for data analysis. In this problem the objective is to decompose a given data matrix $\mathbf{D}$ into the Hadamard product of two low-rank matrices. Given matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$, we consider the decomposition

$$\mathbf{D} = \mathbf{D}_1 \odot \mathbf{D}_2 = (\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2), \tag{4}$$

where $\mathbf{A}_1 \in \mathbb{R}^{n \times h_1}$, $\mathbf{B}_1 \in \mathbb{R}^{h_1 \times m}$, $\mathbf{A}_2 \in \mathbb{R}^{n \times h_2}$ and $\mathbf{B}_2 \in \mathbb{R}^{h_2 \times m}$. Unless stated otherwise, we consider a balanced decomposition with $h_1 = h_2 = h$. Note that unlike in the SVD, no orthogonality constraint is imposed in the problem definition.

A schematic illustration of the decomposition is given in Fig. 3, while Fig. 4 shows an example of Hadamard decomposition in a real-world boolean dataset with $h = 3$. The first low-rank factor exhibits blocks along both diagonals. The second low-rank factor instead has two large blocks along the main diagonal. Since the data are boolean, the Hadamard product corresponds to an element-wise intersection (or logical `and`). As demonstrated in Sect. 8, for this dataset, an equally concise representation obtained via SVD is clearly less accurate than the representation given in Fig. 4.

However, also the Hadamard decomposition in Fig. 4 only provides an approximation to the input matrix $\mathbf{D}$. In general, we are interested in the Hadamard decomposition that best approximates $\mathbf{D}$. Thus, we frame the Hadamard decomposition problem as an optimization problem.

Problem 1 (Hadamard decomposition). Given a matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$ of rank r, and given target ranks h_1 and h_2, find matrices $\hat{\mathbf{D}}_1$ and $\hat{\mathbf{D}}_2$ with ranks at most h_1 and h_2, respectively, so that $\hat{\mathbf{H}} = \hat{\mathbf{D}}_1 \odot \hat{\mathbf{D}}_2$, while minimizing the approximation error:

$$E(\mathbf{D}, \hat{\mathbf{H}}) = \|\mathbf{D} - \hat{\mathbf{H}}\|_F^2. \tag{5}$$

Clearly, for sufficiently large values of h_1 and h_2, it is always possible to obtain $E(\mathbf{D}, \hat{\mathbf{H}}) = 0$: we can have e.g. $\hat{\mathbf{D}}_1 = \mathbf{D}$ and $\hat{\mathbf{D}}_2 = \mathbf{1}$.

Proposition 1 states that if $\hat{\mathbf{D}}_1$ and $\hat{\mathbf{D}}_2$ have rank h_1 and h_2, respectively, then the Hadamard product of $\hat{\mathbf{D}}_1$ and $\hat{\mathbf{D}}_2$ has rank at most $h_1 h_2$. However, there is no guarantee that an arbitrary matrix $\mathbf{D}$ of rank $h_1 h_2$ admits an exact representation as Hadamard decomposition with two factors of rank h_1 and h_2. Generally speaking, finding a globally optimal solution for any choice of h_1 and h_2 may not be feasible.

As mentioned in Sect. 2, for a fixed rank h, it is known that the SVD offers the best possible approximation of $\mathbf{D}$ in terms of squared Frobenius norm. However, the model in Eq. (4) can offer a more accurate description of the input matrix $\mathbf{D}$ compared to the SVD using the same number of basis vectors, since the rank of $\hat{\mathbf{D}}_1 \odot \hat{\mathbf{D}}_2$ can be larger than h. In particular, the Hadamard decomposition represents matrices of rank up to $h_1 h_2$ using $h_1 + h_2$ basis vectors. When $h_1 = h_2 = h$, the Hadamard decomposition can represent exactly some matrices of rank h^2 using $2h$ basis vectors.

On the other hand, when using the same number of basis vectors, the SVD represents with no error matrices of rank only up to $2h$. Hence, the Hadamard decomposition can represent matrices of larger rank than the SVD using the same number of basis vectors whenever $h^2 > 2h$, i.e., $h > 2$. As a consequence, despite the optimality guarantee associated with the SVD, the Hadamard decomposition can outperform the SVD in the task of approximating a matrix $\mathbf{D}$ using the smallest possible amount of information.

The experiments presented in Sect. 8 confirm that the representations provided by the Hadamard decomposition can achieve a more favorable trade-off between accuracy and conciseness compared to the representations provided by SVD, particularly in cases where the potential Hadamard benefit, introduced shortly, is high.

4.1 Relative potential Hadamard benefit

Simple algebra shows that the the value of the error $E(\mathbf{D}, \hat{\mathbf{S}}_k) = \|\mathbf{D} - \hat{\mathbf{S}}_k\|_F^2$ incurred by the rank-k SVD $\hat{\mathbf{S}}_k$ in approximating a rank-r matrix $\mathbf{D}$ can be expressed as the sum of the squares of the singular values from index $k+1$ onward: the error is $\sum_{i=k+1}^{r} \sigma_i^2$. As stated above, the Hadamard decomposition with two rank-h factors can represent exactly some matrices of rank up to h^2. Thus, because of the optimality properties of the SVD, in the best case, such Hadamard decomposition achieves the same approximation error as the rank-h^2 SVD, while using as many basis vectors as the rank-$2h$ SVD.

We define the *relative potential Hadamard benefit* $\gamma(\mathbf{D}, h)$ for the Hadamard decomposition with two rank-h factors as:

$$\gamma(\mathbf{D}, h) = \frac{\sum_{i=2h+1}^{h^2} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}.$$

As a simple example, consider the identity matrix of size 9 and let $h = 3$. Then all singular values are equal to 1, and hence $\gamma(\mathbf{D}, h)$ is 1/3. As the singular values are arranged in non-increasing order, constant singular values result in the maximum values of $\gamma(\mathbf{D}, h)$.

The relative potential Hadamard benefit can be computed easily from the singular values of a matrix $\mathbf{D}$. We show in Sect. 8 that it correlates well with the reduction in approximation error of the Hadamard decomposition over the SVD for the same number of basis vectors. In Appendix A, we investigate in more detail the relationship between the Hadamard decomposition and both the singular values and vectors of a matrix $\mathbf{D}$.

The nonlinearity of the Hadamard product operator makes a thorough theoretical analysis of the Hadamard decomposition model quite challenging. Nonetheless, studying the relationship between the Hadamard decomposition and the SVD provides valuable insights into the Hadamard decomposition model.

4.2 Generalization to multiple factors

We note that Eq. (4) can be extended to model more than two factors, as in: $\mathbf{D} = \bigodot_{i=1}^{q} (\mathbf{A}_i \mathbf{B}_i) = \bigodot_{i=1}^{q} \mathbf{D}_i$.

Assuming the i-th Hadamard factor has rank h_i, the q-factors model can represent matrices of rank up to $h_1 \cdots h_q$ using $2(h_1 + \cdots + h_q)$ basis vectors. Friedenberg et al. (2017) prove that the minimum number of Hadamard factors having rank up to h necessary to represent (without approximation error) any $n \times m$ matrix $\mathbf{D}$ is at

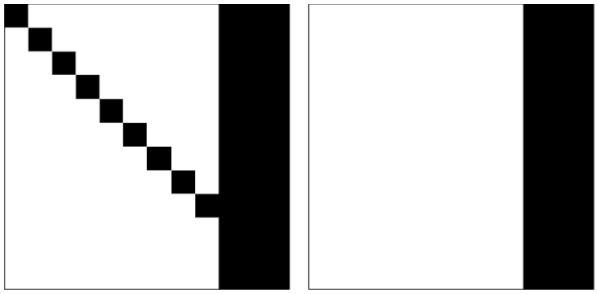

Fig. 5 Left: modified identity matrix with three additional columns with all entries equal to 1. Right: first additive component computed by SVD (after rounding all entries to 0 or 1 using the mean as threshold)

least $\lceil \log_h min(n, m) \rceil$ and at most $\lceil \min(n, m)/(h-1) \rceil$. For simplicity, we henceforth restrict our attention to the two-factor case. A thorough analysis of the model with q factors is left to future work.

4.3 Complexity-theoretic issues

We conclude this section by stating two simple observations regarding the complexity-theoretic status of the Hadamard decomposition problem.

First, consider the variant of Problem 1 where we ask whether the input matrix has a Hadamard decomposition with zero error. This problem can be formulated as a question on the solvability of a set of polynomial equations, and hence it belongs to the complexity class existential theory of the reals (ETR) (Schaefer and Štefankovič 2017).

Second, consider the variant where we look at matrices with entries in $\{0, 1\}$ over the Boolean semiring, where $+$ is interpreted as disjunction and $\times$ as conjunction. Then the existence of a 0-error Hadamard decomposition is trivially in NP: we just guess the decomposition and verify it.

5 Mixed Hadamard decomposition

A matrix may contain some structure that can be described efficiently by the Hadamard decomposition and some structure for which the SVD suits better.

As an artificial example, consider the matrix I' obtained from the 9×9 identity matrix I by concatenating three additional columns with all entries equal to 1; see Fig. 5. In this case, the block of 1s in the last three columns of I' can be modeled exactly by a rank-1 SVD. On the other hand, the diagonal part of the matrix can be captured better by Hadamard decomposition, as we already saw in the example of Fig. 1.

We therefore study a mixed model, combining the SVD with the Hadamard decomposition. Such a model, called mixed Hadamard decomposition, includes a Hadamard product structure while preserving the rank-t SVD, for some integer t, and it is appropriate when a given matrix exhibits submatrices that are best modeled by SVD (i.e., as a sum of few rank-1 matrices) and other submatrices that are

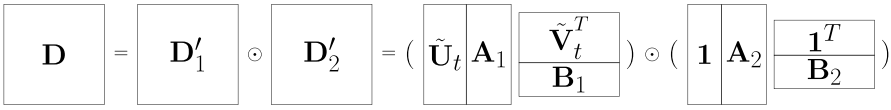

Fig. 6 Diagram illustrating the proposed mixed Hadamard decomposition

best modeled by Hadamard decomposition (i.e., as an element-wise multiplication of two low-rank matrices).

The mixed Hadamard decomposition model builds upon the Hadamard decomposition model $\mathbf{D} = (\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2)$ to incorporate the rank-t SVD. Formally, the mixed Hadamard decomposition model is defined as follows:

$$\mathbf{D} = \mathbf{D}'_1 \odot \mathbf{D}'_2 = \left([\tilde{\mathbf{U}}_t\,;\mathbf{A}_1]\begin{bmatrix}\tilde{\mathbf{V}}_t^T \\ \mathbf{B}_1\end{bmatrix}\right) \odot \left([\mathbf{1}\,;\mathbf{A}_2]\begin{bmatrix}\mathbf{1}^T \\ \mathbf{B}_2\end{bmatrix}\right), \tag{6}$$

where $\tilde{\mathbf{U}}_t = \mathbf{U}_{:,:t}\sqrt{\mathbf{\Sigma}}_{:t,:t}$, $\tilde{\mathbf{V}}_t^T = \sqrt{\mathbf{\Sigma}}_{:t,:t}\mathbf{V}^T_{:t,:}$ and $\mathbf{1}$ denotes the matrix (of suitable dimensions) whose entries are all 1.

Another way of describing the mixed Hadamard decomposition model is to say that it is obtained from the Hadamard decomposition model by constraining the first t columns of $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1^T$ and $\mathbf{B}_2^T$ to match the scaled singular vectors of $\mathbf{D}$ (for $\mathbf{A}_1$ and $\mathbf{B}_1^T$) or to have all entries equal to 1 (for $\mathbf{A}_2$ and $\mathbf{B}_2^T$).

A diagram illustrating the mixed Hadamard decomposition is given in Fig. 6.

Let h_1 and h_2 be the ranks of $\mathbf{D}'_1$ and $\mathbf{D}'_2$, respectively. Then, the larger t is, the lower the upper bound on the rank of the Hadamard product $\mathbf{D}'_1 \odot \mathbf{D}'_2$. For $t = h_1 = h_2$, the mixed Hadamard decomposition coincides with the SVD, while for $t = 0$, the mixed Hadamard decomposition coincides with the Hadamard decomposition. By default, we consider $t = 1$, as empirical evidence suggests that there is rarely any advantage in considering $t \geq 2$.

6 Uniqueness and nonuniqueness

A fundamental question for any matrix decomposition approach is whether there is a *unique decomposition*.

For the case of Hadamard decomposition it is obvious that the decomposition is not unique since we have $\mathbf{D}_1 \odot \mathbf{D}_2 = (\alpha\mathbf{D}_1) \odot (1/\alpha\mathbf{D}_2)$ for arbitrary nonzero α. This can be generalized to any rank-1 matrix $\mathbf{X}$ with nonzero entries. Denote by $\mathbf{X}^{\odot-1}$ the Hadamard (element-wise) inverse of the matrix $\mathbf{X}$. Then if $\mathbf{D} = \mathbf{D}_1 \odot \mathbf{D}_2$, we also have $\mathbf{D} = (\mathbf{D}_1 \odot \mathbf{X}) \odot (\mathbf{X}^{\odot-1} \odot \mathbf{D}_2) = \mathbf{D}_1^* \odot \mathbf{D}_2^*$ where $\mathbf{D}_1^*$ and $\mathbf{D}_2^*$ have the same rank as $\mathbf{D}_1$ and $\mathbf{D}_2$, respectively.

We also have the trivial nonuniqueness from the commutativity of the Hadamard product: $\mathbf{D} = \mathbf{D}_1 \odot \mathbf{D}_2 = \mathbf{D}_2 \odot \mathbf{D}_1$.

This type of "top-level" nonuniqueness aside, we also have a source for nonuniqueness in the second level of the representation: the decomposition $\mathbf{D}_1 = \mathbf{A}_i\mathbf{B}_i$ of the factors of the Hadamard product is not necessarily unique (Piziak and Odell

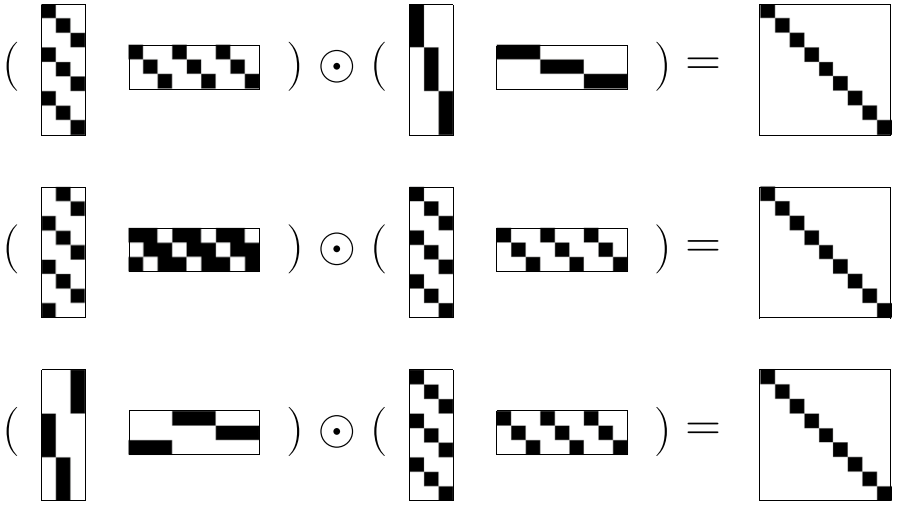

Fig. 7 Example: lack of unique Hadamard decomposition. Three ways of decomposing the 9×9 identity matrix as a Hadamard product of two boolean matrices formed by matrix products with factors of dimensions 9×3 and 3×9

1999; Banerjee and Roy 2014). Intuitively, in the factorization $\mathbf{D}_i = \mathbf{A}_i\mathbf{B}_i$, the columns of $\mathbf{A}_i$ form a basis for the column space of $\mathbf{D}_i$ and $\mathbf{B}_i$ contains the coefficients that express each column of $\mathbf{D}_i$ as a linear combination of the columns of $\mathbf{A}_i$. Therefore, for any basis of the column space of $\mathbf{D}_i$, we have a rank decomposition of the form $\mathbf{D}_i = \mathbf{A}_i\mathbf{B}_i$.

An example of nonuniqueness is illustrated in Fig. 7, where we can see that nonuniqueness is an issue even when considering boolean matrices. The three decompositions in Fig. 7 share a common factor. The factors that differ in the top and middle decompositions in Fig. 7 do not share the same number of 1s.

This observation suggests that introducing in the objective of Problem 1 a regularization (or penalty) term in the form:

$$\|\mathbf{D} - \hat{\mathbf{H}}\|_F^2 + \lambda_1\|\hat{\mathbf{D}}_1\|_F^2 + \lambda_2\|\hat{\mathbf{D}}_2\|_F^2, \tag{7}$$

where $\hat{\mathbf{H}} = \hat{\mathbf{D}}_1 \odot \hat{\mathbf{D}}_2$, may be an effective way to enforce uniqueness. Unfortunately, it is easy to show that such regularization terms are not sufficient for giving unique solutions. This can be seen by considering the *top* and *bottom* decompositions in Fig. 7 where all factors have the same number of 1s.

Other penalty terms could be used to reduce the degree of nonuniqueness. For instance, we could include a penalty to favour orthogonality of the Hadamard factors or even impose constraints to ensure that the Hadamard factors are orthogonal. On the one hand, solutions with orthogonal factors are not only more uniquely defined, but they are (at least sometimes) also more interpretable, since e.g. the columns of an Hadamard factor are uncorrelated among each other and each column tends to be more strongly associated with some columns of the input matrix. On the other hand,

favoring or imposing orthogonality of the Hadamard factors does not guarantee uniqueness and often leads to higher approximation error $E(\mathbf{D}, \hat{\mathbf{H}})$, as we observed in preliminary experiments not reported in this paper. Thus, a more comprehensive examination of the orthogonality-constrained Hadamard decomposition is left to future work.

The example in Fig. 7 uses only boolean factors. Figure 8 gives an example of lack of uniqueness for real-valued factors. There is no obvious way of characterizing the nonuniqueness.

7 Algorithms for the Hadamard decomposition

In this section, we present our method for computing the Hadamard decomposition. Our approach relies on gradient-based optimization.

7.1 Gradient descent for the Hadamard decomposition

For computing the factors in Eq. (4), we give an alternating gradient-descent algorithm where each factor $\mathbf{X} \in \{\mathbf{A}_1, \mathbf{B}_1, \mathbf{A}_2, \mathbf{B}_2\}$ is optimized in turn while fixing the other terms. The pseudo-code of the gradient-descent algorithm is given in Algorithm 1. The matrices $\mathbf{A}_1\,\mathbf{B}_1\,\mathbf{A}_2$ and $\mathbf{B}_2$ are initialized using the SVD of random matrices; by default, we generate for initialization matrices with independent and identically distributed entries drawn from a uniform distribution in [0, 1].

In each iteration (or epoch), matrices $\mathbf{A}_1$, $\mathbf{B}_1$, $\mathbf{A}_2$, and $\mathbf{B}_2$ are updated sequentially by making a step towards the opposite direction of the gradient of $E(\mathbf{D}, \hat{\mathbf{H}}) = \|(\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2) - \mathbf{D})\|_F^2$, computed in matrix form (Petersen et al. 2008). For example, to update matrix $\mathbf{A}_1$, we compute the gradient of the approximation error $E(\mathbf{D}, \hat{\mathbf{H}})$ with respect to $\mathbf{A}_1$. A straightforward application of the chain rule shows that such gradient is $(((\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2) - \mathbf{D}) \odot (\mathbf{A}_2\mathbf{B}_2))\mathbf{B}_1^T$. The gradients of $E(\mathbf{D}, \hat{\mathbf{H}})$ with respect to $\mathbf{B}_1$, $\mathbf{A}_2$ and $\mathbf{B}_2$ are obtained analogously.

The magnitude of the steps in the direction dictated by the gradients of $E(\mathbf{D}, \hat{\mathbf{H}})$ is controlled by the learning rate η.

The gradients of $E(\mathbf{D}, \hat{\mathbf{H}})$ are also multiplied by a $h \times h$ matrix called scaling factor as proposed by Tong et al. (2021); e.g., the scaling factor used when updating $\mathbf{A}_1$ is $(\mathbf{B}_1\mathbf{B}_1^T)^{-1}$. Such scaling factor is not necessary, but it often leads to faster (and more robust) convergence. When scalability is a primary concern, the scaling factor can be omitted since its computation can be significantly time-consuming. Furthermore, the common component of all gradients, i.e., $\Delta = ((\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2) - \mathbf{D})$ is computed once at the beginning of each iteration. In practice, updating Δ prior to each individual matrix update (i.e., four times per iteration) is generally found to increase the runtime and yield only marginal improvements in reconstruction accuracy.

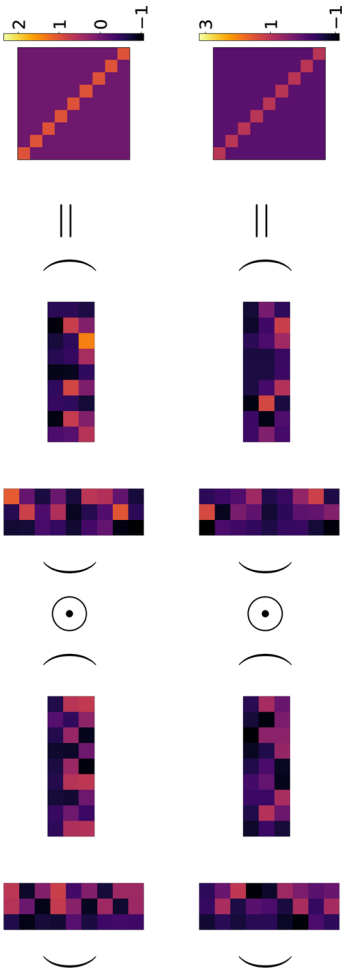

Fig. 8 Example: lack of unique Hadamard decomposition. Two ways of decomposing the 9×9 identity matrix as a Hadamard product of two real-valued matrices formed by matrix products with factors of dimensions 9×3 and 3×9

It is possible that the gradient-descent algorithm fails to converge. In such case, it is appropriate to re-run the algorithm using a smaller learning rate. Our experiments demonstrate that, for a small enough learning rate, the approximation error monotonically decreases. Furthermore, our experiments reveal that by normalizing all the gradients to be of unit norm before each update, divergence becomes unlikely, even for large learning rates. Therefore, by default, we scale gradients to unit norm before each update.

Algorithm 1 is our preferred approach to compute the Hadamard decomposition, and it is used in the experiments presented in Sect. 8. Appendix B discusses briefly some possible alternative algorithms as well as gradient-based algorithms for different (penalized) objective functions and simple extensions of the Hadamard decomposition model in Eq. (4).

Algorithm 1 Scaled gradient descent for Hadamard decomposition

procedure ScaledGradientDescent(D, η, n_i, h)
  $\mathbf{D}_1 \leftarrow \text{RandomMatrix}(n, m)$ ▷ Initialization
  $\mathbf{D}_2 \leftarrow \text{RandomMatrix}(n, m)$
  $\mathbf{U}_1, \mathbf{\Sigma}_1, \mathbf{V}_1^T \leftarrow \text{SVD}(\mathbf{D}_1)$
  $\mathbf{U}_2, \mathbf{\Sigma}_2, \mathbf{V}_2^T \leftarrow \text{SVD}(\mathbf{D}_2)$
  $\mathbf{A}_1, \mathbf{B}_1 \leftarrow \mathbf{U}_1\sqrt{\mathbf{\Sigma}_1}[:, : h], \sqrt{\mathbf{\Sigma}_1}\mathbf{V}_1^T[: h, :]$
  $\mathbf{A}_2, \mathbf{B}_2 \leftarrow \mathbf{U}_2\sqrt{\mathbf{\Sigma}_2}[:, : h], \sqrt{\mathbf{\Sigma}_2}\mathbf{V}_2^T[: h, :]$
  for $i = 1$ to n_i **do**
    $\Delta \leftarrow ((\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2) - \mathbf{D})$
    $\mathbf{A}_1 \leftarrow \mathbf{A}_1 - \eta \cdot ((\Delta \odot (\mathbf{A}_2\mathbf{B}_2))\mathbf{B}_1^T(\mathbf{B}_1\mathbf{B}_1^T)^{-1})$ ▷ Update $\mathbf{A}_1$
    $\mathbf{B}_1 \leftarrow \mathbf{B}_1 - \eta \cdot (((\Delta^T \odot (\mathbf{A}_2\mathbf{B}_2)^T)\mathbf{A}_1(\mathbf{A}_1^T\mathbf{A}_1)^{-1}))^T$ ▷ Update $\mathbf{B}_1$
    $\mathbf{A}_2 \leftarrow \mathbf{A}_2 - \eta \cdot ((\Delta \odot (\mathbf{A}_1\mathbf{B}_1))\mathbf{B}_2^T(\mathbf{B}_2\mathbf{B}_2^T)^{-1})$ ▷ Update $\mathbf{A}_2$
    $\mathbf{B}_2 \leftarrow \mathbf{B}_2 - \eta \cdot (((\Delta^T \odot (\mathbf{A}_1\mathbf{B}_1)^T)\mathbf{A}_2(\mathbf{A}_2^T\mathbf{A}_2)^{-1}))^T$ ▷ Update $\mathbf{B}_2$
  return $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1, \mathbf{B}_2$

7.2 Gradient descent for mixed Hadamard decomposition

We use alternating gradient descent also for computing the mixed Hadamard decomposition. The pseudo-code is given in Algorithm 2. Instead of initializing the matrices randomly, we initialize $\mathbf{A}_1$ and $\mathbf{B}_1$ using the SVD of $\mathbf{D}$ and we initialize $\mathbf{A}_2$ and $\mathbf{B}_2$ randomly, but setting all the entries of the first t columns of $\mathbf{A}_2$ and rows of $\mathbf{B}_2$ to 1. Subsequently, Algorithm 2 proceeds as in Algorithm 1. However, the entries of the first t columns of $\mathbf{A}_1$ and $\mathbf{A}_2$ and the entries of the first t rows of $\mathbf{B}_1$ and $\mathbf{B}_2$ are held constant during the iterations of the gradient-descent procedure. This ensures that the structure of the rank-t SVD is kept intact.

Algorithm 2 Scaled gradient descent for mixed Hadamard decomposition

procedure ScaledGradientDescent-Mixed($\mathbf{D}, \eta, n_i, t, h$)
  $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^T \leftarrow \text{SVD}(\mathbf{D})$
  $\mathbf{D}_1 \leftarrow \mathbf{U}\mathbf{V}^T$ ▷ Initialization
  $\mathbf{D}_2 \leftarrow \text{RandomMatrix}(n, m)$
  $\mathbf{U}_2, \mathbf{\Sigma}_2, \mathbf{V}_2^T \leftarrow \text{SVD}(\mathbf{D}_2)$
  $\mathbf{A}_1, \mathbf{B}_1 \leftarrow \mathbf{U}[:, : h], \mathbf{V}^T[: h, :]$ ▷ Fix additive component
  $\mathbf{A}_2, \mathbf{B}_2 \leftarrow \mathbf{U}_2\sqrt{\mathbf{\Sigma}_2}[:, : h], \sqrt{\mathbf{\Sigma}_2}\mathbf{V}_2^T[: h, :]$
  $\mathbf{A}_2[:, : t] \leftarrow \mathbf{1}$
  $\mathbf{B}_2[: t, :] \leftarrow \mathbf{1}$
  for $i = 1$ to n_i **do**
    $\Delta = ((\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2) - \mathbf{D})$
    $\mathbf{A}_1 \leftarrow \mathbf{A}_1 - \eta \cdot ((\Delta \odot (\mathbf{A}_2\mathbf{B}_2))\mathbf{B}_1^T(\mathbf{B}_1\mathbf{B}_1^T)^{-1})[:, t :]$ ▷ Update $\mathbf{A}_1$
    $\mathbf{B}_1 \leftarrow \mathbf{B}_1 - \eta \cdot (((\Delta^T \odot (\mathbf{A}_2\mathbf{B}_2)^T)\mathbf{A}_1(\mathbf{A}_1^T\mathbf{A}_1)^{-1}))^T[t :, :]$ ▷ Update $\mathbf{B}_1$
    $\mathbf{A}_2 \leftarrow \mathbf{A}_2 - \eta \cdot ((\Delta \odot (\mathbf{A}_1\mathbf{B}_1))\mathbf{B}_2^T(\mathbf{B}_2\mathbf{B}_2^T)^{-1})[:, t :]$ ▷ Update $\mathbf{A}_2$
    $\mathbf{B}_2 \leftarrow \mathbf{B}_2 - \eta \cdot (((\Delta^T \odot (\mathbf{A}_1\mathbf{B}_1)^T)\mathbf{A}_2(\mathbf{A}_2^T\mathbf{A}_2)^{-1}))^T[t :, :]$ ▷ Update $\mathbf{B}_2$
  Return: $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1, \mathbf{B}_2$

7.3 Complexity

The time complexity of the gradient-descent algorithms without the scaling factor for each gradient is $\mathcal{O}(nm \min(n, m) + n_i nhm)$. Similarly, assuming that an $h \times h$ matrix is inverted in time $\mathcal{O}(h^3)$, the time complexity of the scaled gradient-descent algorithm is $\mathcal{O}(nm \min(n, m) + n_i nhm + n_i h^3 + n_i \max(n, m)h^2)$.

The above time-complexity expressions assume that we use methods for matrix multiplication and matrix inversion that require cubic time. In principle, one can resort to algorithms which, given an $n \times n$ matrix, achieve time complexity $\mathbf{O}(n^\omega)$, with $2 < \omega < 3$ (Bürgisser et al. 2013). Furthermore, the scaling factors can be computed efficiently, e.g. by leveraging the SVD, since both the inverse of $(\mathbf{X}\mathbf{X}^T)$ for $\mathbf{X} \in \{\mathbf{B}_1, \mathbf{B}_2\}$ and $(\mathbf{X}^T\mathbf{X})$ for $\mathbf{X} \in \{\mathbf{A}_1, \mathbf{A}_2\}$ can be readily obtained from the SVD of $\mathbf{X}$.

One can also use *stochastic gradient descent* which generally reduces the running time of the algorithms. See Appendix B for details.

Finally, the space complexity of the proposed gradient-descent algorithms is $\mathcal{O}(nm)$, i.e., linear with respect to the size of the input.

8 Experiments

In this section we give experimental results for the Hadamard decomposition approach. Our goal is to assess the performance of the Hadamard decomposition in providing a concise and accurate representation of an input matrix $\mathbf{D}$. We first describe our experimental settings and then our results.

Table 1 Overview of the 250×250 synthetic matrices used in the experiments

Name	Description
U(FR)	Full-rank matrix with entries sampled from a uniform distribution in [0, 1]
G(FR)	Full-rank matrix with entries sampled from a standard Gaussian distribution
ORTHONORMAL	Full-rank matrix obtained from the QR decomposition of G(FR) (Bernstein 2018)
$\mathrm{U_H}$ (**h**)	Hadamard product of two rank-h matrices with entries sampled as in U(FR)
$\mathrm{G_H}$ (**h**)	Hadamard product of two rank-h matrices with entries sampled as in G(FR)
$\mathrm{G_{SVD}}$ (**h**)	Matrix obtained from the rank-h truncated SVD of G(FR)
UT	Matrix obtained from the upper triangular entries of G(FR)
LT	Matrix obtained from the lower triangular entries of G(FR)
BD	Block diagonal matrix with nonzero entries sampled as in U(FR)
BANDED	Boolean matrix exhibiting a banded structure (Garriga et al. 2008)
RECTANGLES	Matrix obtained by generating rectangles of uniformly sampled entries

8.1 Experimental setting

We describe the datasets we use, the baselines, the performance-evaluation metrics, and the parameter choices.

8.1.1 Datasets

We consider both synthetic and real-world datasets. Synthetic matrices help us to understand better how different data characteristics impact the performance of the Hadamard decomposition method, while the real-world datasets are chosen to reflect data characteristics arising in different application domains. The synthetic matrices all are of dimensions 250×250. They include full-rank random matrices with entries drawn from uniform (U(FR)) and Gaussian (G(FR)) distributions, random matrices generated as the Hadamard product of two low-rank random uniform ($\mathrm{U_H}(h)$) and Gaussian ($\mathrm{G_H}(h)$) factors, orthonormal matrices (ORTHONORMAL), low-rank matrices obtained from the truncated SVD of a Guassian matrix ($\mathrm{G_{SVD}}(h)$), banded matrices (BANDED) (Garriga et al. 2008), upper triangular matrices (UT) and lower triangular matrices (LT). A brief description of the synthetic datasets is given in Table 1.

For real-world datasets, we consider matrices from different domains: CAMERAMAN, LENA, CAT, DOG, OLIVETTI, and DIGITS are images; LESMISERABLES and FOOTBALL are graph adjacency matrices; MOVIETWEETINGS, MOVIELENS-100K, and YELP are ratings data; and the remaining datasets, PALEO, GENES, ARRHYTHMIA and SPECTROMETER, contain matrices from other domains.

Table 2 Summary characteristics and references for the real-world datasets used in the experiments

Name	Rows	Columns	Density	Data type	References
LesMiserables	77	77	0.086	Boolean	Knuth (1993)
Football	115	115	0.093	Boolean	Girvan and Newman (2002)
Paleo	124	139	0.115	Boolean	Fortelius (2008)
Spectrometer	531	101	0.997	Real	Center (1988)
Arrhythmia	452	120	0.675	Real	Guvenir et al. (1997)
Lena	256	256	1.000	Integer	Rodriguez-Aragon and Zhigljavsky (2010)
cameraman	256	256	1.000	Integer	Lee et al. (2005)
Yelp	1 449	135	0.048	Boolean	Sajnani et al. (2012)
Dog	399	600	0.988	Integer	Ozbulak (2017)
Cat	400	600	0.987	Integer	Ozbulak (2017)
Genes	7 129	38	0.997	Integer	Brunet et al. (2004)
Digits	2 000	240	0.607	Integer	Alpaydin and Kaynak (1998)
MovieTweetings	1 000	1 000	0.002	Integer	Dooms et al. (2013)
MovieLens-100k	943	1 609	0.039	Integer	Harper and Konstan (2015)
Olivetti	400	4 096	0.999	Real	Chung et al. (1999)

Summary characteristics and references for all the real-world datasets used in our experimental analysis are given in Table 2. Furthermore, more details on both the real and the synthetic datasets used in our experiments are given in Appendix C.

8.1.2 Metrics

Let $\hat{\mathbf{D}}$ represent the output of any algorithm $\mathcal{A}$ for input $\mathbf{D}$. Unless specified otherwise, we measure the performance of algorithm $\mathcal{A}$ using the approximation error:

$$E(\mathbf{D}, \hat{\mathbf{D}}) = \|\mathbf{D} - \hat{\mathbf{D}}\|_F^2. \tag{8}$$

8.1.3 Baselines

Since we are evaluating our methods using the approximation error, and since the SVD can be analytically shown to minimize the approximation error in Eq. (8) for a given rank (Golub and Van Loan 2013), we compare our method only against SVD.

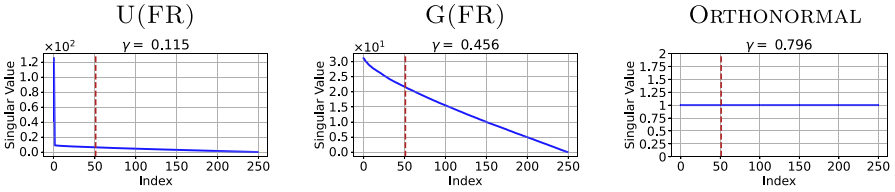

Fig. 9 Singular values and relative potential Hadamard benefit γ in synthetic synthetic datasets of dimensions 250×250. The blue line shows all the singular values σ_i, from the largest to the smallest. For each dataset, the relative potential Hadamard benefit γ for a balanced Hadamard decomposition with factors of rank $h = 25$ is reported. For reference, a red dashed vertical line is added in correspondence of $2h = 50$. The value of γ is obtained as the ratio of the sum of squared singular values to the right of the red dashed line to the sum of all squared singular values

8.1.4 Parameters

We consider *balanced* Hadamard decompositions with two factors $\mathbf{D}_1$ and $\mathbf{D}_2$ of rank h, which are compared with the SVD of rank $2h$. The value h is varied in $\{4, 6, 9, 13, 20, 30, 40\}$, except in the two real-world datasets of smallest rank where only smaller values of h are appropriate. Finally, the parameter t in the mixed Hadamard decomposition is always set to 1.

8.1.5 Implementation

Our Python implementation is available online.[4] The NUMBA library (Lam et al. 2015) is used for optimization. Experiments are performed on a computer with 2×10 core Xeon E5 processor and 256 GB memory. All reported results are averages over 10 runs.

8.2 Results

We first present results on three example matrices, which highlight important aspects of the Hadamard decomposition, and then we present more extensive results comparing the Hadamard decomposition and the SVD on both synthetic and real-world datasets.

Illustrative example of results in synthetic datasets: the full-rank uniform, Gaussian and orthonormal matrices. The properties of the Hadamard decomposition over the SVD vary greatly over matrices with different characteristics; the relative potential Hadamard benefit γ is a useful tool in understanding when and why the Hadamard approach works well. We illustrate this by considering the 250×250 full-rank synthetic uniform (U(FR)), Gaussian (G(FR)) and orthonormal (ORTHONORMAL) matrices.

Figure 9 shows the decay of singular values in such matrices, and gives the relative potential Hadamard benefit γ for a balanced Hadamard decomposition with both factors of rank $h = 25$.

[4] https://github.com/maciap/HadamardDecomposition.

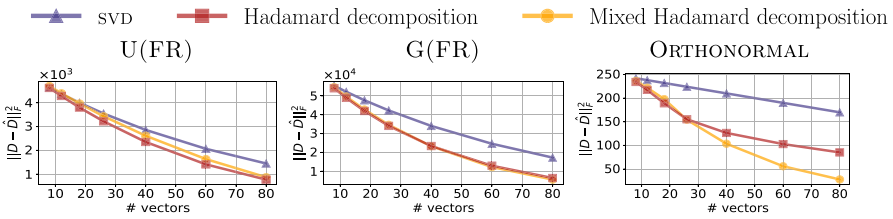

Fig. 10 Experiments on the full-rank uniform, Gaussian and orthonormal matrices. Approximation error $E(\mathbf{D}, \hat{\mathbf{D}})$ incurred by the rank-$2h$ SVD as well as by the Hadamard decomposition and the mixed Hadamard decomposition, both with two rank-h factors, as a function of $2h$, i.e., the number of basis vectors

For the uniform random matrix U(FR) the singular values decay steeply. Therefore the value of γ is small, and the potential usefulness of the Hadamard decomposition is limited. On the contrary, any orthonormal matrix has constant singular values. This is the scenario where the Hadamard decomposition achieves the highest potential benefit. Finally, the random standard Gaussian matrix G(FR) represents an intermediate scenario.

Figure 10 shows the approximation error achieved by SVD, Hadamard decomposition and mixed Hadamard decomposition on the U(FR), G(FR) and ORTHONORMAL matrices as a function of the number of basis vectors used for reconstruction. The results show that the empirical benefits of the Hadamard decomposition are indeed predicted by the potential Hadamard benefit.

8.2.1 Results on other synthetic datasets

Figure 11 gives the results analogous to those shown in Fig. 10 for the other synthetic matrices used in our experiments.

First, consider the $\mathrm{G}_{\mathrm{SVD}}$ **(15)** matrix. This matrix is the rank-15 truncated SVD of G(FR), and it serves as an example of a matrix where the SVD produces better results than the Hadamard decomposition. In this scenario, by definition, the SVD only needs 15 basis vectors to achieve 0 approximation error. All but the first 15 singular values of the $\mathrm{G}_{\mathrm{SVD}}$ **(15)** matrix are 0, and hence the relative potential Hadamard benefit γ rapidly decreases to 0 as h approaches 8. Thus, we again see that the distribution of the singular values provides important information about the usefulness of the Hadamard decomposition.

When looking at results for the U_{H} **(10)**, U_{H} **(25)** and U_{H} **(50)** matrices, which are obtained as Hadamard products of two rank-constrained uniform matrices, we note that the advantage of the Hadamard decomposition over the SVD is limited. The reason is that, again, the singular values of these matrices decay steeply and thus the potential Hadamard benefit is small. For instance, the relative potential Hadamard benefit γ in the U_{H} **(25)** matrix is of the order of 10^{-5}, whereas in the full-rank U(FR) matrix γ is of the order of 10^{-1}.

The G_{H} **(10)**, G_{H} **(25)** and G_{H} **(50)** matrices, produced as the Hadamard product of two rank-constrained standard Gaussian matrices, depict a different scenario: as

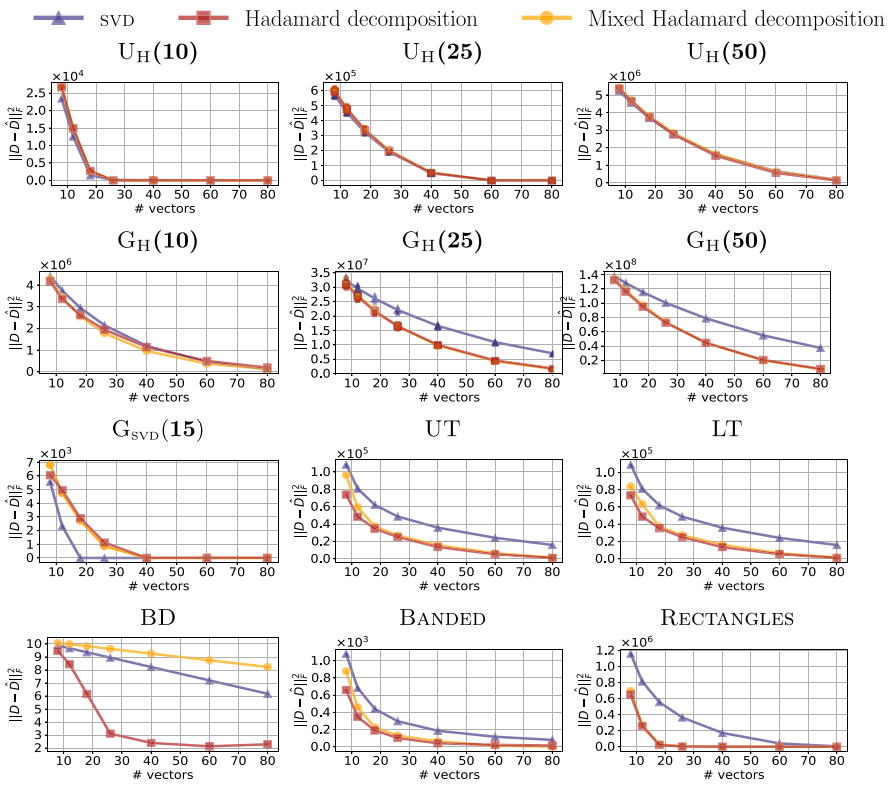

Fig. 11 Experiments on synthetic data. Approximation error $E(\mathbf{D}, \hat{\mathbf{D}})$ incurred by the rank-$2h$ SVD as well as by the Hadamard decomposition and the mixed Hadamard decomposition, both with two rank-h factors, as a function of $2h$, i.e., the number of basis vectors

the number of basis vectors used for reconstruction increases, the advantage of the Hadamard decomposition over the SVD becomes clear.

Clear advantages of the Hadamard decomposition over the SVD are also observed in the remaining synthetic datasets: the upper (UT) and lower (LT) triangular matrices, the block-diagonal matrix (BD), the banded matrix (BANDED) and the matrix with rectangles of nonzero entries (RECTANGLES).

Finally, the performance of the mixed Hadamard decomposition is often in between that of the Hadamard decomposition and the SVD. In some cases, however, the mixed Hadamard decomposition achieves lower approximation error than both Hadamard decomposition and SVD.

8.2.2 Results on real-world datasets

Figure 12 summarizes the results on the real-world datasets. While the experiments on real data generally lead to similar findings as the experiments on synthetic data, they also provide new insights.

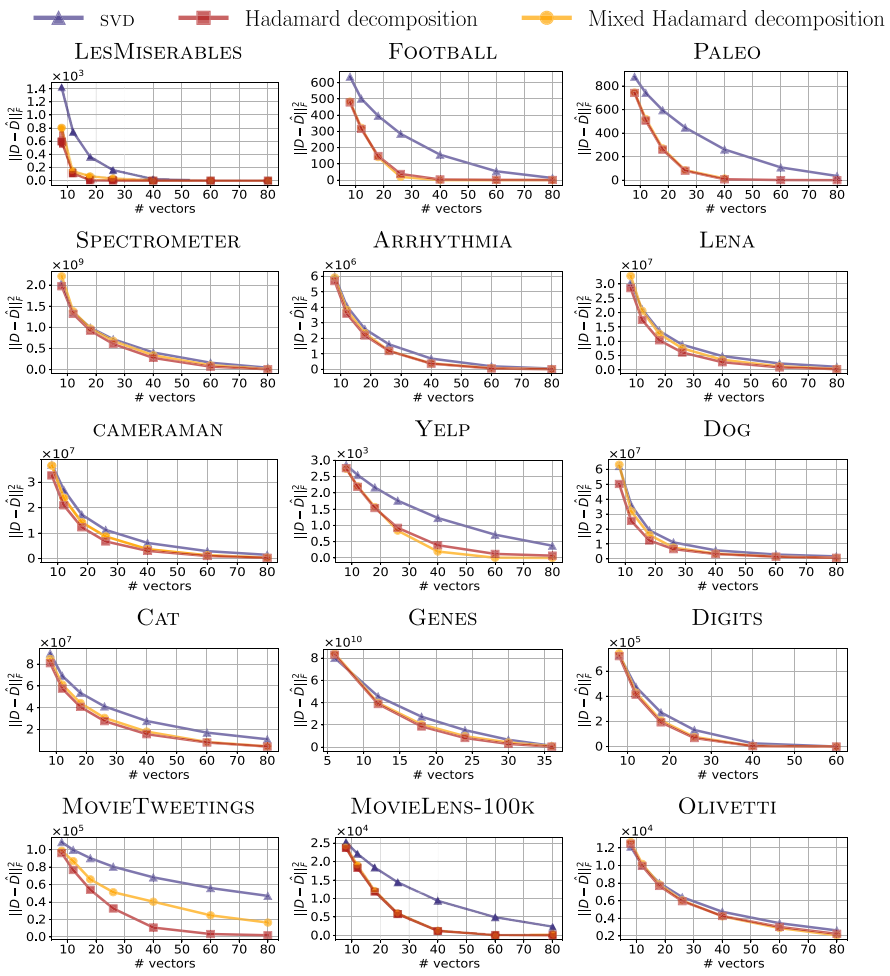

Fig. 12 Experiments on real data. Approximation error $E(\mathbf{D}, \hat{\mathbf{D}})$ incurred by the rank-$2h$ SVD as well as by the Hadamard decomposition and the mixed Hadamard decomposition, both with two rank-h factors, as a function of $2h$, i.e., the number of basis vectors

For instance, in the PALEO dataset, which is a binary matrix with approximately banded structure (Fortelius 2008), the Hadamard decomposition results in even higher benefits than in the synthetic banded matrix (Fig. 11). The benefits of the Hadamard decomposition over the SVD are also evident in the MOVIELENS-100K, MOVIETWEETINGS and YELP datasets, pointing towards possible advantages of the Hadamard decomposition in generating recommendations. We address this theme in more detail in Sect. 8.3. Similarly, advantages of the Hadamard decomposition over the SVD are also significant in image datasets, indicating that the Hadamard decomposition can provide a more accurate image compression than the SVD at parity compression ratio.

8.2.3 Variability in the results

We conclude the present section with a note on the variability in the results.

In the experiments with synthetic data, there are two sources of randomness: the data generation and the initialization of the gradient-descent algorithm. For the experiments with real data, only the the initialization of the gradient descent injects randomness in the results.

Similarly, there are two forms of variability in the results: either only the approximation accuracy can vary between runs, or both the approximation accuracy and the retrieved Hadamard decomposition.

It turns out that the variance in the approximation accuracy across multiple runs is rather small. In experiments on both synthetic and on real data, we find that the gradient descent is quite stable and randomness does not have a significant impact on the accuracy of the results of the gradient descent. For instance, across all the reported experimental results with U(FR) and G(FR), the average coefficient of variation[5] of the approximation error $E(\mathbf{D}, \hat{\mathbf{D}})$ for the Hadamard decomposition is approximately 0.005. Therefore, in all the presented experiments, we decide to report averages over 10 repetitions and we do not show the (minimal) variability in the results.

The variation in the resulting basis vectors is harder to quantify. As discussed in Sect. 6, there are many possible forms of nonuniqueness for the Hadamard decomposition and there may exist several highly different Hadamard decompositions with approximately the same accuracy.

8.3 Case study—application to recommender systems

In recommender systems (Hallinan and Striphas 2016; Ramlatchan et al. 2018) the objective is to suggest items to users based on their preferences. A widely used approach to build recommender systems is collaborative filtering (Ramlatchan et al. 2018), which seeks to predict the preferences of a user by leveraging information from a group of users with similar tastes. Matrix decomposition techniques, such as the SVD, are highly popular in (model-based) collaborative filtering.

Specifically, in collaborative filtering, a modified SVD is computed which is able to account for missing ratings as well as for the tendency of some users to rate items higher than others, or the tendency of some items to be rated higher than others (Funk 2006). The resulting algorithm, henceforth SVD-CF, is learned via an highly scalable stochastic gradient-descent algorithm. This algorithm can be adapted to the Hadamard decomposition in order to learn the Hadamard-CF. The details of the SVD-CF and the Hadamard-CF algorithms are described in Appendix D.1.

SVD-CF and Hadamard-CF only account for explicit ratings (i.e., numerical ratings). The extension of SVD-CF to also account for implicit ratings, known in the literature as SVD++, provides more competitive performance in model-based collaborative filtering (Wang et al. 2020). While Hadamard-CF can also be extended to account for implicit ratings, in this section our interest lies in comparing the

[5] Ratio of the standard deviation to the mean.

Table 3 Experiments with recommender systems. Root mean squared error in the test set in three datasets (MovieLens-100k, MovieLens-1M and Netflix) for svd-CF and Hadamard-CF

Algorithm	Dataset		
	Netflix	MovieLens-1 M	MovieLens-100k
SVD-CF	0.308	0.287	0.296
Hadamard-CF	0.247	0.228	0.237

performance of svd-CF and Hadamard-CF, and we leave extensions accounting for implicit ratings to future work.

Table 3 presents a comparison of svd-CF and Hadamard-CF, demonstrating the advantages of a matrix decomposition based on the Hadamard product in the context of recommender systems. The comparison is based on the test set root mean squared error, a popular metric used to assess the performance of recommender systems. Definitions and more details concerning the experiments to evaluate Hadamard-CF against svd-CF are given in Appendix D.2.

9 Conclusions and open problems

We have introduced the Hadamard decomposition problem in data analysis, where the task is to find an approximate representation of the input matrix as a Hadamard product of two (or more) low-rank matrices. The motivation for the problem comes from situations where the input data can be assumed to have some type of multiplicative structure.

We have shown that this decomposition method has the potential for expressing some matrices more succinctly than the svd. While the Hadamard decomposition is not unique, our experiments show that a gradient-descent method generally yields good results, giving competitive or better approximations of matrices than the svd with the same number of basis vectors. In particular, for several real-world datasets we consider, the Hadamard decomposition achieves comparable approximation accuracy with the svd, but in some other datasets, the Hadamard decomposition offers a clearly more accurate approximation, hinting at underlying multiplicative structure on those datasets. To determine when the proposed Hadamard decomposition should be preferred to the svd, we have introduced the relative potential Hadamard benefit, a simple measure which can be obtained from the singular values of a matrix. If for a matrix the relative potential Hadamard benefit is high, the Hadamard decomposition is anticipated to lead to a considerably higher approximation accuracy than the svd with the same number of basis vectors.

We have also described a mixed decomposition model which combines the Hadamard decomposition with the svd. This model is designed for decomposing matrices where the top-level structure is an svd-like additive one, and then there is a Hadamard product of low-rank matrices.

There are many open problems in the study of the Hadamard decomposition. Among them, it would be valuable to obtain a better understanding of the forms of nonuniqueness of the decomposition. The experiments show that the Hadamard decomposition often produces smaller approximation error than the SVD. On the other hand, SVD has the advantage that it normally produces a unique answer and that it is faster to compute. Having some characterization of the nonuniqueness would alleviate the first issue.

For the algorithmic part, the proposed method is a straightforward application of gradient descent, and it would be very interesting to develop a faster algorithm or a combinatorial algorithm which ideally would always find an optimal solution when one exists. Furthermore, it would be interesting to study the extension of the Hadamard decomposition to other types of matrices, such as non-negative and Boolean. The formal computational complexity of the Hadamard decomposition problem is also open.

With respect to applications, it would be valuable to investigate the benefits of the Hadamard decomposition for different machine-learning and data-mining tasks, beyond collaborative filtering. For instance, the Hadamard decomposition could be used as a pre-processing step for classification and clustering tasks (Singh and Leavline 2019). Further, since matrix decomposition and clustering are closely related (Ding et al. 2005; Li and Ding 2013), extending the principles of the Hadamard decomposition to the clustering setting is an interesting direction for future research.

Finally, because of the nonlinearity and lack of uniqueness of the Hadamard decomposition, the results of the method are not necessarily easy to interpret. Still, the results of the Hadamard decomposition on the paleontological dataset (Fortelius et al. 2006; Fortelius 2008) are very promising, as they seem to capture important parts of the underlying phenomenon. It is therefore an interesting question to interpret the multiplicative structure visible in the results, and to assess the effect of the nonuniqueness of the results on interpretability.

Appendix A. Relationship with the SVD

In this section, we present some connections between SVD and the Hadamard decomposition. These results do not lead to an algorithm to compute the Hadamard decomposition. However, they provide valuable insights into the Hadamard decomposition.

First, we describe simple error bounds for the Hadamard decomposition that follow from the properties of the SVD. Then, we discuss other interesting connections between SVD and the Hadamard decomposition.

A.1. Error bounds

As discussed in Sect. 4 the analysis of the spectrum of singular values of a matrix $\mathbf{D}$ leads to the notion of relative potential Hadamard benefit, a useful measure for

predicting the gain that can be achieved by adopting the Hadamard decomposition in place of the SVD. A similar measure, the unnormalized potential Hadamard benefit is given by the numerator of the relative potential Hadamard benefit $\gamma(\mathbf{D}, h)$. More specifically, the *potential Hadamard benefit* $\Gamma(\mathbf{D}, h)$ when using Hadamard decomposition with two rank-h factors is the sum of squared singular values σ_i^2, for $i = 2h + 1$ to h^2:

$$\Gamma(\mathbf{D}, h) = \sum_{i=2h+1}^{h^2} \sigma_i^2.$$

From the definition of potential Hadamard benefit and the optimality properties of SVD, we have the following.

Proposition 2 *Given a matrix* $\mathbf{D}$, *let* $\hat{\mathbf{H}}_h$ *be the estimate of* $\mathbf{D}$ *given by an optimal Hadamard decomposition with both factors of rank up to* h, *and let* $\hat{S}_k$ *be the estimate of* $\mathbf{D}$ *given by the truncated SVD of rank* k. *Denote* $E(\mathbf{D}, \hat{\mathbf{H}}_h) = \|\mathbf{D} - \hat{\mathbf{H}}_h\|_F^2$ *the approximation error incurred by the Hadamard decomposition with two rank-*h *factors and denote* $E(\mathbf{D}, \hat{\mathbf{S}}_k) = \|\mathbf{D} - \hat{\mathbf{S}}_k\|_F^2$ *the approximation error incurred by the SVD truncated to* k *basis vectors*. *Then we have*:

$$\text{(i) } E(\mathbf{D}, \hat{\mathbf{H}}_h) \geq E(\mathbf{D}, \hat{\mathbf{S}}_{h^2}) \quad \text{and} \quad \text{(ii) } E(\mathbf{D}, \hat{\mathbf{S}}_{2h}) - E(\mathbf{D}, \hat{\mathbf{H}}_h) \leq \Gamma(\mathbf{D}, h).$$

Proof As regards inequality (i), note that the Hadamard decomposition reconstructs matrices of rank up to h^2. As explained in Sect. 4, the lowest possible approximation error incurred by any matrix of rank h^2 is achieved by the truncated SVD of rank h^2 (Stewart 1993), from which inequality (i) follows. Instead, inequality (ii) directly follows from the definition of potential Hadamard benefit. □

It is also interesting to notice that a loose upper bound on $E(\mathbf{D}, \hat{\mathbf{H}})$, independent of the singular values, is given by: $E(\mathbf{D}, \hat{\mathbf{H}}_h) \leq E(\mathbf{D}, \hat{\mathbf{S}}_h)$, which is trivially obtained by taking $\mathbf{D}_1$ to be the rank-h SVD and $\mathbf{D}_2$ to be the rank-1 matrix whose entries are all equal to 1. As demonstrated in our experiments, this trivial upper bound is remarkably loose, as in practice the Hadamard decomposition with two rank-h factors often outperforms the rank-$2h$ SVD. On the other hand, it is possible to demonstrate that the lower bound (i) in Proposition 2 can be achieved. For instance, as shown in Sect. 1, the Hadamard decomposition with two factors of rank 3 reconstructs exactly the 9×9 identity matrix using 6 basis vectors, so that $E(\mathbf{D}, \hat{\mathbf{H}}_h) = 0$. To achieve the same error, the SVD needs 9 basis vectors. Thus, in this example, it holds that $E(\mathbf{D}, \hat{\mathbf{H}}_h) = E(\mathbf{D}, \hat{\mathbf{S}}_{h^2})$.

It is important to highlight that, as the proposed gradient-based algorithms are not guaranteed to produce optimal results, the bounds presented above, although insightful, may not apply in practice.

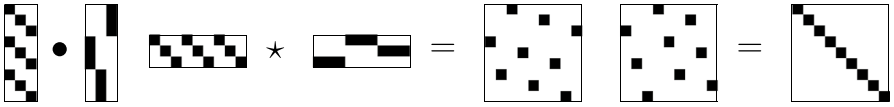

Fig. 13 Example: alternative formulation of the Hadamard decomposition using the row-wise and column-wise Kronecker products. Black squares represent 1 entries

A.2. Use of SVD in finding a Hadamard decomposition

For the results presented in the remainder of this section, we introduce some additional matrix-product operators.

Given two matrices $\mathbf{A}$ and $\mathbf{B}$ of dimensions $n_A \times m_A$ and $n_B \times m_B$, respectively, their Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is defined as a block matrix of dimensions $n_A n_B \times m_A m_B$, consisting of $n_A \times m_A$ blocks, each of dimensions $n_B \times m_B$, so that its block indexed by i and j is $[\mathbf{A}]_{i,j}\mathbf{B}$. We also define the row-wise and column-wise Kronecker product operators (Khatri and Rao 1968). For matrices $\mathbf{A}$ and $\mathbf{B}$ with dimensions $n_A \times m$ and $n_B \times m$, their column-wise Kronecker product $\mathbf{A} \star \mathbf{B}$ is a $n_A n_B \times m$-matrix whose rows are the Hadamard (element-wise) products of each pair of rows $\mathbf{a}^T$ and $\mathbf{b}^T$, such that $\mathbf{a}^T$ is a row of $\mathbf{A}$ and $\mathbf{b}^T$ is a row of $\mathbf{B}$. The row-wise Kronecker product is defined similarly. For matrices $\mathbf{A}$ and $\mathbf{B}$ with dimensions $n \times m_A$ and $n \times m_B$, their row-wise Kronecker product $\mathbf{A} \bullet \mathbf{B}$ is an $(n \times m_A m_B)$-matrix whose columns are the Hadamard products of each pair of columns $\mathbf{a}$ and $\mathbf{b}$, such that $\mathbf{a}$ is a column of $\mathbf{A}$ and $\mathbf{b}$ is a column of $\mathbf{B}$.

Several problems in linear algebra admit a solution based on the SVD. For instance, the solution to the problem of finding the nearest orthogonal matrix to a given matrix $\mathbf{D} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}$ is given by $\mathbf{U}\mathbf{V}^T$ (Schönemann 1966). As another example, the problem of finding the (rank-unconstrained) Kronecker product $\mathbf{X} \otimes \mathbf{Y}$ that is the closest to an input matrix $\mathbf{D}$ (Van Loan and Pitsianis 1993) admits a closed-form solution in terms of the SVD of a permuted version of $\mathbf{D}$. Thus, it is natural to ask whether a solution to Problem 1 could also be found using an SVD-based approach. In the remainder of this section, we present some possible ideas in this direction; however, while they shed light on the properties of the Hadamard decomposition, they do not lead to algorithms that would improve on the gradient-descent methods.

First, we provide an alternative formulation for the Hadamard decomposition, which exposes more clearly the connection between Hadamard decomposition and SVD. In particular, as also shown in Fig. 13, we can re-write $\mathbf{D} = (\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2)$ as the (standard) matrix product of two factors, as follows:

$$\mathbf{D} = (\mathbf{A}_1\mathbf{B}_1) \odot (\mathbf{A}_2\mathbf{B}_2) = (\mathbf{A}_1 \bullet \mathbf{A}_2)(\mathbf{B}_1 \star \mathbf{B}_2) = \mathbf{A}\mathbf{B}. \tag{9}$$

Equation (9) is a consequence of known properties connecting the different product operators (Slyusar 1999), namely:

$$(\mathbf{A}\mathbf{C}) \odot (\mathbf{C}\mathbf{D}) = (\mathbf{A} \bullet \mathbf{B})(\mathbf{C} \star \mathbf{D}). \tag{10}$$

If the dimensions of $\mathbf{A}_1$ and $\mathbf{A}_2$ are $n \times h$, and the dimensions of $\mathbf{B}_1$ and $\mathbf{B}_2$ are $h \times m$, then the dimensions of $\mathbf{A}$ and $\mathbf{B}$ are $n \times h^2$ and $h^2 \times m$. The singular value decomposition $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ also provides a similar decomposition $\mathbf{D} = \mathbf{A}_{\text{svd}}\mathbf{B}_{\text{svd}}$, with $\mathbf{A}_{\text{svd}} = \mathbf{U}\sqrt{\mathbf{\Sigma}}$ and $\mathbf{B}_{\text{svd}} = \sqrt{\mathbf{\Sigma}}\mathbf{V}^T$. Although in special cases it may hold that $\mathbf{A} = \mathbf{A}_{\text{svd}}$ and $\mathbf{B} = \mathbf{B}_{\text{svd}}$, in general, this is not the case. For instance, the columns of $\mathbf{A}$ and $\mathbf{B}$ are typically not orthogonal, unlike the columns of $\mathbf{A}_{\text{svd}}$ and $\mathbf{B}_{\text{svd}}$. However, the following result holds in general.

Proposition 3 *Let* $\mathbf{D}$ *be a rank-*h^2 *matrix obtained as the Hadamard product of two rank-*h *factors. Using the above notation, we have the following.*

(i) $\mathbf{B}$ *and* $\mathbf{B}_{\text{svd}}$ *span the same row space.*
(ii) $\mathbf{A}$ *and* $\mathbf{A}_{\text{svd}}$ *span the same column space.*

Proof To prove that $\mathbf{B}$ and $\mathbf{B}_{\text{svd}}$ span the same row space and $\mathbf{A}$ and $\mathbf{A}_{\text{svd}}$ span the same column space, we show that the rows of $\mathbf{B}_{\text{svd}}$ are linear combinations of the rows of $\mathbf{B}$ and similarly for the columns of $\mathbf{A}_{\text{svd}}$ and $\mathbf{A}$. Since we consider the usual balanced Hadamard decomposition with two rank-h factors, we have $\mathbf{A}, \mathbf{A}_{\text{svd}} \in \mathbb{R}^{n \times h^2}$ and $\mathbf{B}, \mathbf{B}_{\text{svd}} \in \mathbb{R}^{h^2 \times m}$. Thus, $\mathbf{A}$ and $\mathbf{A}_{\text{svd}}$ both contain h^2 linearly independent columns and likewise $\mathbf{B}$ and $\mathbf{B}_{\text{svd}}$ both contain h^2 linearly independent rows. We have:

$$\mathbf{AB} + \mathbf{A}_{\text{svd}}\mathbf{B}_{\text{svd}} = \mathbf{D} + \mathbf{D} = 2\mathbf{D}.$$

Multiplying each entry of a matrix by a constant does not affect the rank of the matrix (Bernstein 2018). Hence, the rank of $2\mathbf{D}$ is equivalent to the rank of $\mathbf{D}$, which is h^2. Furthermore, linear algebra shows that we also have the following factorization of $2\mathbf{D}$:

$$2\mathbf{D} = \mathbf{AB} + \mathbf{A}_{\text{svd}}\mathbf{B}_{\text{svd}} = [\mathbf{A}\,;\mathbf{A}_{\text{svd}}] \begin{bmatrix} \mathbf{B} \\ \mathbf{B}_{\text{svd}} \end{bmatrix},$$

where $[\mathbf{A};\mathbf{A}_{\text{svd}}]$ denotes column-wise concatenation and $\begin{bmatrix} \mathbf{B} \\ \mathbf{B}_{\text{svd}} \end{bmatrix}$ denotes row-wise concatenation. We conclude that, since $2\mathbf{D}$ has rank h^2, $[\mathbf{A}\,;\mathbf{A}_{\text{svd}}]$ can have up to h^2 linearly independent columns and $\begin{bmatrix} \mathbf{B} \\ \mathbf{B}_{\text{svd}} \end{bmatrix}$ can have up to h^2 linearly independent rows, and because both $\mathbf{A}$ and $\mathbf{A}_{\text{svd}}$ have h^2 linearly independent columns and similarly both $\mathbf{B}$ and $\mathbf{B}_{\text{svd}}$ have h^2 linearly independent rows, the claim follows. □

Proposition 3 exposes the theoretical underpinnings of the Hadamard decomposition, suggesting that columns of $\mathbf{A}_1$ and $\mathbf{A}_2$, like the left singular vectors, lead to a basis for the column space of $\mathbf{D}$ and $\mathbf{B}_1$ and $\mathbf{B}_2$, like the right singular vectors, lead to a basis for the row space of $\mathbf{D}$. Unlike the SVD, the bases identified by the Hadamard decomposition are not orthogonal. However, $\mathbf{A}$ is

related to $\mathbf{A}_{\text{svd}}$ through a simple change of basis. Analogously, $\mathbf{B}$ is related to $\mathbf{B}_{\text{svd}}$ through a simple change of basis.

This also suggests that the building blocks of the Hadamard decomposition, $\mathbf{A}_1$, $\mathbf{B}_1$, $\mathbf{A}_2$ and $\mathbf{B}_2$, are to be searched within the span of the left and right singular vectors of $\mathbf{D}$. In practice, however, the Hadamard decomposition model may not hold exactly and it is not convenient to restrict the search carried out via gradient descent to the span of the left and right singular vectors.

A.3. SVD of a Hadamard product

In the previous paragraph, we have considered using the SVD as a starting point for finding the Hadamard decomposition of a matrix $\mathbf{D}$. Another possibility is to consider the relationship between the SVD of a Hadamard product and the SVDs of its factors, and how this could lead to an efficient algorithm for estimating the Hadamard decomposition. Let $\mathbf{D} = \mathbf{D}_1 \odot \mathbf{D}_2$, $\mathbf{D}_1 = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T$ and $\mathbf{D}_2 = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T$. The following equality holds:

$$\mathbf{D} = (\mathbf{U}_1 \bullet \mathbf{U}_2)(\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma_2})(\mathbf{V}_1^T \star \mathbf{V}_2^T). \tag{11}$$

To prove Eq. (11), we again rely on the properties of the different product operators considered in this section. In particular, we have:

$$\begin{aligned}
\mathbf{D} &= \mathbf{D}_1 \odot \mathbf{D}_2 \\
&= (\mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T) \odot (\mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T) \\
&= (\mathbf{U}_1 \bullet \mathbf{U}_2)(\mathbf{\Sigma}_1\mathbf{V}_1^T \star \mathbf{\Sigma}_2\mathbf{V}_2^T) \\
&= (\mathbf{U}_1 \bullet \mathbf{U}_2)(\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma}_2)(\mathbf{V}_1^T \star \mathbf{V}_2^T),
\end{aligned}$$

where we have used Eq. (10) as well as another known equality (Rao 1970):

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \star \mathbf{D}) = (\mathbf{AC}) \star (\mathbf{BD}). \tag{12}$$

The decomposition in Eq. (11) does not correspond to the SVD of $\mathbf{D}$ in any obvious way. Further, it does not correspond to the singular value decomposition of any matrix since e.g. $(\mathbf{U}_1 \bullet \mathbf{U}_2)$ and $(\mathbf{V}_1^T \star \mathbf{V}_2^T)$ are not orthogonal matrices. The lack of a standard relationship between the SVD of an Hadamard product and that of its factors motivates the search of a solution to the Hadamard decomposition problem using gradient-based methods rather than methods based on linear algebra or combinatorial algorithms.

Appendix B. Alternative approaches for Hadamard decomposition

In this section, we discuss alternative gradient-based approaches for the Hadamard decomposition problem, as well as simple modifications of the proposed gradient-descent algorithm to accommodate basic extensions of the Hadamard decomposition model.

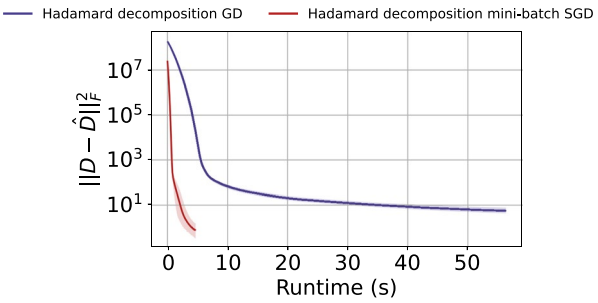

Fig. 14 PALEO dataset. Approximation error $E(\mathbf{D}, \hat{\mathbf{D}})$ as a function of runtime (in seconds) for gradient descent (GD) and mini-batch stochastic gradient descent (mini-batch SGD) to compute the Hadamard decomposition. The figure displays the average $E(\mathbf{D}, \hat{\mathbf{D}})$ across 30 repetitions as well as shaded regions corresponding to 95% confidence intervals

B.1. Stochastic gradient descent

The proposed gradient-descent algorithms compute, at each iteration, the gradient of the approximation error with respect to the matrices $\mathbf{A}_1$, $\mathbf{B}_1$, $\mathbf{A}_2$, and $\mathbf{B}_2$. Since computing gradients involves multiple matrix multiplications, the proposed gradient-descent methods become slow as the matrix dimensions n, m and the ranks of the Hadamard factors h_1 and h_2 grow.

It is possible to modify the proposed gradient-descent algorithms so that a subset of rows of suitable size $1 \leq b < n$ is considered to take a step. For $b = 1$, the resulting algorithms are referred to as stochastic gradient descent, while for $1 < b < n$ the resulting algorithms are referred to as mini-batch stochastic gradient descent.

To implement (mini-batch) stochastic gradient descent, it is sufficient to modify the proposed algorithms (e.g., Algorithm 1) so that only up to b rows of $\mathbf{D}$, $\mathbf{A}_1$ and $\mathbf{A}_2$ are considered for each update. This avoids all multiplications of large matrices, thus greatly reducing the computational burden.

To demonstrate the efficiency gain obtained by mini-batch stochastic gradient descent for computing the Hadamard decomposition, Fig. 14 illustrates the approximation error $E(\mathbf{D}, \hat{\mathbf{D}})$ incurred by both Algorithm 1 and its mini-batch stochastic-gradient-descent variant as a function of runtime for the PALEO dataset. Here, both Hadamard factors have rank 40 and the mini-batch size b for mini-batch stochastic gradient descent is set to 16. Figure 14 suggests that even in a dataset of moderate size, mini-batch stochastic gradient descent can lead to convergence much earlier than Algorithm 1, although, as indicated by the confidence intervals, the decay of $E(\mathbf{D}, \hat{\mathbf{D}})$ for mini-batch stochastic gradient descent exhibits significant fluctuations over time.

B.2. Coordinate descent

Stochastic gradient descent computes gradients associated with subsets of the entries of a matrix. It is also possible to take this idea further and design a gradient-descent

algorithm where each entry of each matrix $\mathbf{X}_{i,j}$, with $\mathbf{X} \in \{\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1, \mathbf{B}_2\}$, is updated in turn.

Calculating gradients with respect to each specific entry in every matrix is simple, and it is also feasible to derive formulas for closed-form updates by setting gradients to 0. For instance, the formula for updating the entry indexed by i and j of A_1, obtained by setting to 0 the associated gradient, is:

$$[\mathbf{A}_1]_{i,j} \leftarrow \sum_{p=1}^{m}([\mathbf{B}_1]_{j,p}[\mathbf{D}_2]_{i,p}([\mathbf{D}]_{i,p} - \sum_{s=1,s\neq j}^{h}([\mathbf{A}_1]_{i,s}[\mathbf{B}_1]_{s,p}[\mathbf{D}_2]_{i,p}))) / \sum_{p=1}^{m}([\mathbf{B}_1]_{j,p}[\mathbf{D}_2]_{i,p})^2,$$

and similarly for the other entries and the other matrices involved in the gradient-descent algorithm.

The ensuing algorithm, dubbed *coordinate descent*, at each iteration, updates all entries of each matrix in sequence. Alternatively, it is straightforward to convert the coordinate-descent algorithm into the more scalable stochastic coordinate-descent algorithm, which updates a randomly drawn entry at each iteration.

B.3. Penalized gradient descent

As mentioned in Sect. 6, including a penalty term in the objective function to be minimized via gradient descent can lead to a decrease in the degree of nonuniqueness.

For instance, a small modification to the update steps in Algorithm 1 suffices to minimize the following regularized objective function:

$$\|\mathbf{D} - \hat{\mathbf{H}}\|_F^2 + \lambda_1\|\hat{\mathbf{D}}_1\|_F^2 + \lambda_2\|\hat{\mathbf{D}}_2\|_F^2. \tag{13}$$

The gradient involved in the update of $\mathbf{A}_1$ is modified as follows:

$$\Delta \odot (\mathbf{A}_2\mathbf{B}_2)\mathbf{B}_1^T + \lambda_1(\mathbf{A}_1\mathbf{B}_1)\mathbf{B}_1^T,$$

and similarly for $\mathbf{A}_2$, $\mathbf{B}_1$ and $\mathbf{B}_2$. The same scaling factor as in Algorithm 1 (e.g., $(\mathbf{B}_1\mathbf{B}_1^T)^{-1}$ in the update of $\mathbf{A}_1$) can also be used.

Moreover, frameworks for automatic differentiation, such as PyTorch (Paszke et al. 2017), allow to easily perform gradient descent while incorporating arbitrary penalty terms in the objective function, even in cases where the gradients are not analytically computable.

B.4. Gradient descent for more general functions

In some cases, we may be interested in a model of the form:

$$\mathbf{D} = f(\mathbf{D}_1 \odot \mathbf{D}_2), \tag{14}$$

where f is an arbitrary element-wise function. For example, if $\mathbf{D}$ is a matrix with values ranging from 0 to 1, setting f to be the sigmoid function (i.e., $f = (1 + e^{-x})^{-1}$) guarantees that the approximation $f(\mathbf{D}_1 \odot \mathbf{D}_2)$ consists of values in the same range as $\mathbf{D}$ and can improve the quality of the approximation.

It is simple to extend the proposed gradient-descent algorithms to the model in Eq. (14) where f is the sigmoid function. For example, using the chain rule, the gradient involved in the update of $\mathbf{A}_1$ is modified as follows:

$$\frac{e^{-(\mathbf{A}_1\mathbf{B}_1)\odot(\mathbf{A}_2\mathbf{B}_2)}}{(e^{-(\mathbf{A}_1\mathbf{B}_1)\odot(\mathbf{A}_2\mathbf{B}_2)}+1)^2}(\mathbf{D} - \frac{1}{1+e^{-(\mathbf{A}_1\mathbf{B}_1)\odot(\mathbf{A}_2\mathbf{B}_2)}}) \odot (\mathbf{A}_2\mathbf{B}_2)\mathbf{B}_1^T,$$

and similarly for $\mathbf{A}_2$, $\mathbf{B}_1$ and $\mathbf{B}_2$.

As for penalized gradient descent, automatic-differentiation tools allow to easily carry out gradient descent for arbitrary functions f without the need for manually deriving analytical gradients.

Appendix C. Detailed description of datasets

In this section, we provide a detailed description of the datasets used in our experimental evaluation.

C.1. Synthetic datasets

We consider 15 synthetic types of matrices. All synthetic matrices have 250 rows and columns. The data-generating mechanism associated with each synthetic matrix is described next.

- U(FR): full-rank matrix with i.i.d. entries drawn from a uniform distribution in [0, 1].
- G(FR): full-rank matrix with i.i.d. entries drawn from a standard Gaussian distribution.
- ORTHONORMAL: full-rank matrix obtained from the QR decomposition of G(FR) (Bernstein 2018).
- $\mathrm{U_H}(\mathbf{h})$: low-rank matrix obtained as the Hadamard product of two rank-h matrices (i.e., according to the Hadamard decomposition) with independent and identically distributed (i.i.d.) entries drawn from a uniform distribution in [0, 1]. We show results for $h \in \{10, 25, 50\}$.
- $\mathrm{G_H}(\mathbf{h})$: low-rank matrix obtained as the Hadamard product of two rank-h matrices (i.e., according to the Hadamard decomposition) with i.i.d. entries drawn from a standard Gaussian distribution. We show results for $h \in \{10, 25, 50\}$.
- $\mathrm{G_{SVD}}(\mathbf{h})$: low-rank matrix obtained as the rank-h truncated SVD of a matrix with i.i.d. entries drawn from a standard Gaussian distribution. In other words, this

matrix is obtained from the SVD of G(FR) by setting to 0 all but the highest h singular values.
- UT: full-rank triangular matrix obtained from the upper triangular entries of G(FR).
- LT: full-rank triangular matrix obtained from the lower triangular entries of G(FR).
- BD: block diagonal matrix with $w = 200$ blocks of random size along the main diagonal with i.i.d entries drawn from a uniform distribution in [0, 1].
- BANDED: boolean matrix (with entries in $\{0, 1\}$), exhibiting a banded structure (Garriga et al. 2008), which is obtained by starting with a matrix of all 0s and setting to 1 the j-th entry in the i-th row if $|i - j| \leq w$, where we set $w = 50$.
- RECTANGLES: matrix with $w = 50$ blocks (tiles) of i.i.d. entries drawn from a uniform in [0, 100]. The starting point of each block is sampled uniformly at random among all entries and the support in the rows and the columns is sampled according to a uniform distribution in [0, 12]. In addition, we add a large rectangle with constant entries accounting for a fixed proportion of entries (30% by default).

C.2. Real-world datasets

In addition to 15 synthetic datasets, we consider 15 heterogeneous real-world datasets. Real-world datasets include graph adjancency matrices (LESMISERABLES and FOOTBALL), images (CAMERAMAN, LENA, CAT, DOG, OLIVETTI, and DIGITS) and ratings (MOVIETWEETINGS, MOVIELENS-100K, and YELP). In addition, the PALEO dataset contains information about locality and species of mammal fossils. The SPECTROMETER dataset contains a number of spectrometer measurements. The ARRHYTHMIA dataset pertains to the medical domain and contains several attributes useful to distinguish between the presence and absence of cardiac arrhythmia. Finally, the biological GENES dataset is used for molecular classification of cancer by gene expression monitoring.

Appendix D. Details of application to recommender systems

In Sect. 8.3 we briefly describe an application of the ideas underlying the Hadamard decomposition to recommender systems. Here, we provide more details related to such application. Specifically, we first describe the details of the SVD-CF and Hadamard-CF algorithms and then we discuss the details of the experiments performed to assess the performance of SVD-CF and Hadamard-CF.

D.1. Algorithms

Recommender systems are ubiquitous nowadays. For instance, they are used by *E-commerce platforms*, *online social media* and *streaming services* (Hallinan and Striphas 2016; Ramlatchan et al. 2018).

The goal of recommender systems is to predict the ratings that a set of users will give to a set of items (or products).

The impact of the SVD in the field of recommender systems is best introduced starting from a famous case study. In 2006, Netflix, an american company offering *on-demand video streaming service*, promised a one-milion-dollar prize to anyone who could improve over the current recommender systems by at least 10% in *root mean squared error (RSME)* (Bennett et al. 2007), a popular performance metric defined later in this section.

The winning approach for the Netflix prize was based on SVD-CF and since the Netflix price, SVD-CF has experienced a notable increase in popularity in the field of research studying recommender systems (Guan et al. 2017; Zhou et al. 2015).

In applications to recommender systems, the SVD generates a number of latent factors representing users and items.

In practice, not all users rate all items. The SVD as described in Sect. 3 is not able to account for missing ratings. Thus, in the context of recommender systems, the SVD is computed while taking into account only the ratings that are available. More specifically, the available rating $r_{u,i}$ for user u and item i is modelled as follows:

$$\hat{r}_{u,i} = q_i^T p_u, \tag{15}$$

where q_i is a vector representing the affinity of the i-th item for each of the latent factors, and similarly, p_u is a vector representing the affinity of the u-th user for each of the latent factors. The vectors q_i and p_u for all i and u can be learned by applying stochastic gradient descent to minimize: $\sum_{r_{u,i} \in R_{train}} (r_{u,i} - q_i^T p_u)^2$, where R_{train} is the set of available ratings. This approach has an important drawback. Some users tend to give significantly higher ratings than others. Similarly, some products may generally be rated higher than others. Thus, more accurate recommendations are obtained via the following model:

$$\hat{r}_{u,i} = \mu + b_u + b_i + q_i^T p_u, \tag{16}$$

where μ is the global average of all the ratings, whereas b_u and b_i are biases given by the user-specific and item-specific average ratings, respectively.

Moreover, it is often the case that the latent factors have to generalize to unseen data R_{test}. Therefore, it is found to be effective to include a regularization term, so that the objective function to be minimized becomes as follows:

$$\sum_{r_{u,i} \in R_{train}} (r_{u,i} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda (b_u^2 + b_i^2 + \|q_i\|^2 + \|p_u\|^2), \tag{17}$$

where λ controls the amount of regularization.

All the parameters (i.e., biases μ, b_u and b_i as well as vectors q_i and p_u for all i and u) can be learned via stochastic gradient descent. Stochastic gradient descent iterates through each available rating in R_{train} and updates all the involved parameters in the directions of the negative gradients of the regularized squared error in Eq. (17).

The model assumed by Eq. (17), learned via stochastic gradient descent, is referred to as SVD-CF.

SVD-CF represents ratings as (translated) dot products computed in the latent space identified by the SVD. Assume that items represent movies. Further, imagine that one latent factor represents, say, the level of action, while another latent factor represents the popularity of the actors. Some users may be interested exclusively in movies that have a high level of action and also have a popular cast of actors, but they may not be interested at all in movies that have a high level of action and no popular actors.

The dot product in the latent space computed in the SVD-CF model fails to accurately capture the preferences of similar users and, for those users, it may predict a significant rating for a movie that has a high level of action but does not have a popular cast of actors.

On the other hand hand, the product of two latent-space dot products can capture more naturally the preferences of users who are solely interested in movies that have both a high level of action and popular actors.

Thus, SVD-CF can be extended using the Hadamard product, resulting in the Hadamard-CF model, which minimizes the following objective function:

$$\begin{aligned}\sum_{r_{u,i}\in R_{train}} &(r_{u,i} - \mu - b_u - b_i - (q_{i_1}^T p_{u_1})(q_{i_2}^T p_{u_2}))^2 \\ &+ \lambda(b_u^2 + b_i^2 + \|q_{i_1}\|^2 + \|p_{u_1}\|^2 + \|q_{i_2}\|^2 + \|p_{u_2}\|^2).\end{aligned} \tag{18}$$

Also the parameters of the Hadamard-CF model are learned via stochastic gradient descent. For the same reason why the Hadamard decomposition may outperform the SVD in reconstruction accuracy while using the same number of parameters, Hadamard-CF may outperform SVD-CF.

D.2. Experiments

In Sect. 8.3, we present experiments demonstrating that the Hadamard-CF algorithm can generate more accurate recommendations than SVD-CF. We consider the popular MOVIELENS-100K (Harper and Konstan 2015), MOVIELENS-1M (Harper and Konstan 2015) and NETFLIX (Bennett et al. 2007) benchmark datasets. MOVIELENS-100K contains 100, 000 ratings, whereas MOVIELENS-1M and NETFLIX contain 1 milion ratings. Ratings in all datasets are normalized in the range [0, 1]. The 75% of the rating data is used as a train set R_{train} and the remaining data R_{test} are dedicated to testing.

In order to compare the biased SVD model (SVD-CF) and the Hadamard-product-based biased Hadamard decomposition model (Hadamard-CF), we apply weak regularization, setting $\lambda = 10^{-3}$, and we investigate performance, measured by the standard root mean squared error measure, computed in the test set, while ensuring the two approaches under comparison rely on the same amount of parameters (both biases and latent factors). The test set *RMSE* is defined by:

$$\sqrt{\frac{1}{n_r}\sum_{r_{u,i}\in R_{test}}(r_{u,i}-\hat{r}_{u,i})^2}, \tag{19}$$

where n_r is the total number of ratings in R_{test} and $\hat{r}_{u,i}$ represents the prediction for rating $r_{u,i}$.

The user-specified number of latent factors is varied in [80, 160, 200, 320, 480]. For each number of factors, we run 10 experiments and Table 3 in Sect. 8.3 reports the average *RMSE* over all runs.

Acknowledgements The authors would like to thank Petteri Kaski for many valuable discussions and Kaie Kubjas for pointing out reference (Oneto and Vannieuwenhoven 2023) to us.

Author contributions HM and AG first discussed the basic idea of the Hadamard decomposition problem. MC, AG and HM studied the problem. MC and HM implemented the algorithms and carried out experiments. MC, AG and HM contributed to writing and reviewing all sections.

Funding Open Access funding provided by Aalto University.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no Conflict of interest.

References

Banerjee S, Roy A (2014) Linear algebra and matrix analysis for statistics. CRC Press, Boca Raton

Bernstein D (2018) Scalar, vector, and matrix mathematics: theory, facts, and formulas-revised and expanded edition, Expanded. Princeton University Press, Princeton

Breslow NE, Lubin J, Marek P, Langholz B (1983) Multiplicative models and cohort analysis. J Am Stat Assoc 78(381):1–12

Bro R, Smilde AK (2014) Principal component analysis. Ana Methods 6(9):2812–2831

Brunet J-P, Tamayo P, Golub TR, Mesirov JP (2004) Metagenes and molecular pattern discovery using matrix factorization. Proc Natl Acad Sci 101(12):4164–4169

Bürgisser P, Clausen M, Shokrollahi MA (2013) Algebraic complexity theory, vol 315. Springer, New York

De Handschutter P, Gillis N, Siebert X (2021) A survey on deep matrix factorizations. Comput Sci Rev 42:100423

DeSantis D, Skau E, Truong DP, Alexandrov B (2022) Factorization of binary matrices: rank relations, uniqueness and model selection of Boolean decomposition. ACM Trans Knowl Discov Data (TKDD) 16(6):1–24

Fortelius M, Gionis A, Mannila H, Jernvall J (2006) Spectral ordering and biochronology of European fossil mammals. Paleobiology 32(2):206–214
Gillis N (2020) Nonnegative matrix factorization. SIAM, Philadelphia
Girvan M, Newman ME (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826
Golub GH, Van Loan CF (2013) Matrix computations. JHU Press, Baltimore
Guan X, Li C-T, Guan Y (2017) Matrix factorization with rating completion: an enhanced SVD model for collaborative filtering recommender systems. IEEE Access 5:27668–27678
Hallinan B, Striphas T (2016) Recommended for you: the Netflix prize and the production of algorithmic culture. New Media Soc 18(1):117–137
Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst (TIIS) 5(4):1–19
Hoff PD (2017) Lasso, fractional norm and structured sparse estimation using a Hadamard product parametrization. Computat Stat Data Anal 115:186–198
Hoff P (2021) Additive and multiplicative effects network models. Stat Sci 36(1):34–50
Horn RA, Yang Z (2020) Rank of a Hadamard product. Linear Algebra Appl 591:87–98
Klema V, Laub A (1980) The singular value decomposition: its computation and some applications. IEEE Trans Autom Control 25(2):164–176
Knuth DE (1993) The Stanford GraphBase: a platform for combinatorial computing, vol 1. ACM Press, New York
Lee P (2001) Relation between exposure to asbestos and smoking jointly and the risk of lung cancer. Occup Environ Med 58(3):145–153
Lee C-S, Guo S-M, Hsu C-Y (2005) Genetic-based fuzzy image filter and its application to image processing. IEEE Trans Syst Man Cybern Part B (Cybern) 35(4):694–711
Liu R, Li S, Liu J, Ma L, Fan X, Luo Z (2020) Learning Hadamard-product-propagation for image dehazing and beyond. IEEE Trans Circuits Syst Video Technol 31(4):1366–1379
Miettinen P, Vreeken J (2014) Mdl4bmf: minimum description length for Boolean matrix factorization. ACM Trans Knowl Discov Data (TKDD) 8(4):1–31
Miettinen P, Mielikäinen T, Gionis A, Das G, Mannila H (2008) The discrete basis problem. IEEE Trans Knowl Data Eng 20(10):1348–1362
Petersen KB, Pedersen MS et al (2008) The matrix cookbook. Tech Univ Den 7(15):510
Piziak R, Odell P (1999) Full rank factorization of matrices. Math Mag 72(3):193–201
Qi L, Cornelis MC, Zhang C, Van Dam RM, Hu FB (2009) Genetic predisposition, western dietary pattern, and the risk of type 2 diabetes in men. Am J Clin Nutr 89(5):1453–1458
Ramlatchan A, Yang M, Liu Q, Li M, Wang J, Li Y (2018) A survey of matrix completion methods for recommendation systems. Big Data Min Anal 1(4):308–323
Rao CR (1970) Estimation of heteroscedastic variances in linear models. J Am Stat Assoc 65(329):161–172
Roberto Cruz M (1992) More about the multiplicative model for the analysis of genotype-environment interaction. Heredity 68(2):135–140
Rodriguez-Aragon LJ, Zhigljavsky A (2010) Singular spectrum analysis for image processing. Stat Interface 3(3):419–426
Schaefer M, Štefankovič D (2017) Fixed points, nash equilibria, and the existential theory of the reals. Theory Comput Syst 60(2):172–193
Schönemann PH (1966) A generalized solution of the orthogonal procrustes problem. Psychometrika 31(1):1–10
Singh DAAG, Leavline EJ (2019) Dimensionality reduction for classification and clustering. Int J Intell Syst Appl (IJISA) 11(4):61–68
Slyusar V (1999) A family of face products of matrices and its properties. Cybern Syst Anal 35(3):379–384
Stewart GW (1993) On the early history of the singular value decomposition. SIAM Rev 35(4):551–566
Styan GP (1973) Hadamard products and multivariate statistical analysis. Linear Algebra Appl 6:217–240
Thompson JN (1999) The evolution of species interactions. Science 284(5423):2116–2118
Thompson PL, MacLennan MM, Vinebrooke RD (2018) Species interactions cause non-additive effects of multiple environmental stressors on communities. Ecosphere 9(11):02518
Tong T, Ma C, Chi Y (2021) Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent. J Mach Learn Res 22(1):6639–6701

Udell M, Townsend A (2019) Why are big data matrices approximately low rank? SIAM J Math Data Sci 1(1):144–160. https://doi.org/10.1137/18M1183480
Ursu E, Duchesne P (2009) On multiplicative seasonal modelling for vector time series. Stat Probab Lett 79(19):2045–2052
Van Loan CF, Pitsianis N (1993) Approximation with Kronecker products. In: Moonen MS, Golub GH (eds) Linear algebra for large scale and real time applications. Kluwer Publications, Dordrecht
Visick G (2000) A quantitative version of the observation that the Hadamard product is a principal submatrix of the Kronecker product. Linear Algebra Appl 304(1–3):45–68
Wang S, Sun G, Li Y (2020) Svd++ recommendation algorithm based on backtracking. Information 11(7):369
Yang Z, Stoica P, Tang J (2019) Source resolvability of spatial-smoothing-based subspace methods: a Hadamard product perspective. IEEE Trans Signal Process 67(10):2543–2553
Ye T, Du SS (2021) Global convergence of gradient descent for asymmetric low-rank matrix factorization. Adv Neural Inf Process Syst 34:1429–1439
Zhou X, He J, Huang G, Zhang Y (2015) SVD-based incremental approaches for recommender systems. J Comput Syst Sci 81(4):717–733
Alpaydin E, Kaynak C (1998) Optical recognition of handwritten digits. UCI machine learning repository. https://doi.org/10.24432/C50P49
Bennett J, Lanning S et al (2007) The Netflix prize. In: Proceedings of KDD cup and workshop, vol. 2007, p 35
Center NAR (1988) Low resolution spectrometer. UCI machine learning repository. https://doi.org/10.24432/C5B02R
Chung K-C, Kee SC, Kim SR (1999) Face recognition using principal component analysis of Gabor filter responses. In: Proceedings international workshop on recognition, analysis, and tracking of faces and gestures in real-time systems. In conjunction with ICCV'99 (Cat. No. PR00378). IEEE, pp 53–57
Cox DR (1984) Interaction. International Statistical Review/Revue Internationale de Statistique 1–24
Ding C, He X, Simon HD (2005) On the equivalence of nonnegative matrix factorization and spectral clustering. In: Proceedings of the 2005 SIAM international conference on data mining. SIAM, pp 606–610
Dooms S, De Pessemier T, Martens L (2013) Movietweetings: a movie rating dataset collected from twitter. In: Workshop on Crowdsourcing and human computation for recommender systems, vol 2013. CrowdRec at RecSys, p 43
Fortelius M (2008) Neogene of the old world database of fossil mammals (NOW). http://www.helsinki.fi/science/now/
Friedenberg N, Oneto A, Williams RL (2017) Minkowski sums and Hadamard products of algebraic varieties. In: Combinatorial algebraic geometry: selected papers from the 2016 Apprenticeship Program, pp 133–157
Funk S (2006) Netflix update: try this at home
Garriga GC, Junttila E, Mannila H (2008) Banded structure in binary matrices. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 292–300
Guvenir HA, Acar B, Demiroz G, Cekin A (1997) A supervised machine learning algorithm for arrhythmia analysis. In: Computers in cardiology 1997. IEEE, pp 433–436
Horn RA (1990) The Hadamard product. In: Proc. symp. appl. math., vol 40, pp 87–169
Hyeon-Woo N, Ye-Bin M, Oh T-H (2021) FedPara: low-rank Hadamard product for communication-efficient federated learning. In: International conference on learning representations
Khatri C, Rao CR (1968) Solutions to some functional equations and their applications to characterization of probability distributions. Sankhyā Indian J Stat Ser A 167–180
Kim J-H, On K-W, Lim W, Kim J, Ha J-W, Zhang B-T (2016) Hadamard product for low-rank bilinear pooling. In: International conference on learning representations
Lam SK, Pitrou A, Seibert S (2015) Numba: a llvm-based python jit compiler. In: Proceedings of the second workshop on the LLVM compiler infrastructure in HPC, pp 1–6
Li T, Ding CH (2013) Nonnegative matrix factorizations for clustering: a survey. In: Data clustering: algorithms and applications, pp 149–176
Li Y, Liang Y (2017) Provable alternating gradient descent for non-negative matrix factorization with strong correlations. In: International conference on machine learning. PMLR, pp 2062–2070
Mnih A, Salakhutdinov RR (2007) Probabilistic matrix factorization. Adv Neural Inf Process Syst 20

Montúfar G (2018) Restricted Boltzmann machines: introduction and review. In: Information geometry and its applications: on the occasion of Shun-ichi Amari's 80th Birthday, IGAIA IV Liblice, Czech Republic, June 2016. Springer, pp 75–115

Oneto A, Vannieuwenhoven N (2023) Hadamard-Hitchcock decompositions: identifiability and computation. arXiv preprint arXiv:2308.06597

Ozbulak U (2017) Singular value decomposition on images. GitHub

Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in pytorch. In: NIPS-W

Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747

Sajnani H, Saini V, Kumar K, Gabrielova E, Choudary P, Lopes C (2012) Classifying Yelp reviews into relevant categories. Mondego Group, Univ. California Press, Berkeley, CA USA, Tech. Rep

Salakhutdinov R, Mnih A (2008) Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th international conference on machine learning, pp 880–887

Wu CW (2018) Prodsumnet: reducing model parameters in deep neural networks via product-of-sums matrix decompositions. arXiv preprint arXiv:1809.02209