Information Sciences 621 (2023) 562–579

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Elastic adversarial deep nonnegative matrix factorization for matrix completion

Seyed Amjad Seyedi, Fardin Akhlaghian Tab [*], Abdulrahman Lotfi, Navid Salahian, Jovan Chavoshinejad

Department of Computer Engineering, University of Kurdistan, Sanandaj, Iran

A R T I C L E   I N F O

Article history:
Received 23 May 2022
Received in revised form 20 November 2022
Accepted 24 November 2022
Available online 1 December 2022

Keywords:
Deep learning
Adversarial learning
Generalization
Elastic loss

A B S T R A C T

In recent years, matrix completion has attracted a lot of attention. Conventional matrix completions are shallow and ineffective when dealing with complex data structures. Researchers have recently tried to incorporate deep structures into matrix completion; however, considerable challenges still exist. Most matrix completion methods may fail to work effectively in the presence of limited observations. To enhance the generalization of the reconstruction, adversarial methods are proposed that attempt to fool models by providing deceptive input. The aim is to develop an adversarial training algorithm that resists attacks in a deep model, thus at the same time leading to enhancing the generalization. Therefore, in this paper, we propose an elastic adversarial training to design a high-capacity Deep Nonnegative Matrix Factorization (DNMF) model with proper discovery latent structure of the data and enhanced generalization abilities. In other words, the challenges mentioned above are addressed by perturbing the inputs in DNMF with an elastic loss which is intercalated and adapted between Frobenius and $\ell_{2,1}$ norms. This model not only dispenses with adversarial DNMF generation but also is robust towards a mixture of multiple attacks to attain improved accuracy. Extensive simulations show that the proposed approach outperforms state-of-the-art methods.

1. Introduction

Matrix completion (MC) is recovering or predicting missing or unknown items in partially seen matrices. It has found widespread use in applications such as collaborative filtering [1], image inpainting [2,3], and data classification [4,5]. Most real-world data (e.g., images and user ratings) are low-rank or nearly low-rank. Therefore, most existing matrix completion problems are often solved by considering the low-rank prior and reducing the rank of the underlying matrix. These approaches could be divided into two main categories: Nuclear-Norm Minimization (NNM) and Matrix Factorization (MF) [6]. Due to the non-convexity and discontinuous character of rank minimization, this problem is generally NP-hard. Therefore, the rank minimization problem cannot be solved exactly using the available techniques. A convex relaxation for matrix rank is the nuclear-norm, defined as the sum of singular values.

* Corresponding author.
E-mail addresses: amjadseyedi@uok.ac.ir (S.A. Seyedi), f.akhlaghian@uok.ac.ir (F. Akhlaghian Tab), a.lotfi@uok.ac.ir (A. Lotfi), n.salahian@uok.ac.ir (N. Salahian), j.chavoshinejad@uok.ac.ir (J. Chavoshinejad).

https://doi.org/10.1016/j.ins.2022.11.120
0020-0255/© 2022 Elsevier Inc. All rights reserved.

Therefore, Matrix completion techniques based on nuclear-norm minimization have been suggested [7,8]. The singular value thresholding (SVT) technique [9], the Inexact Augmented Lagrange Multiplier (IALM) method [10], and the Alternating Direction Method (ADM) [8] have all been used to solve nuclear-norm minimization. Recent modifications or enhancements to nuclear-norm have been used for matrix completion. For example, Truncated Nuclear-Norm (TNN) reduction is proposed in [2] for matrix completion. For matrix rank, TNN is a better approximation than the nuclear-norm. The rationale is that the largest few singular values generally include essential information and should be maintained; however, small ones should be reduced.

Low-rank matrix completion is another name for conventional matrix completion, which assumes that the provided incomplete matrix has a low-rank structure. Matrix factorization can be used to complete a low-rank matrix, where the incomplete matrix is approximated by multiplying a basis matrix and a coefficient matrix. The matrix rank must be known in advance in the matrix factorization-based techniques [11]. For factorization-based matrix completion, a method known as the low-rank matrix fitting algorithm (LMaFit) was introduced in [12] to modify the matrix rank adaptively. Furthermore, tensor factorization and completion have gained prominence with the growing use of streaming tensor data such as videos and audios. In contrast to matricization methods that use nuclear-norm or matrix factorization, tensor completion methods generalize matrix-form data to high-dimensional tensors. [13–16].

Since the main computation of matrix factorization-based techniques is the multiplication of a basis matrix and a coefficient matrix in each iteration, they have low computational complexity. They are, however, non-convex, and their performance is affected by the given or estimated rank. Although adapting the nuclear-norm minimization criterion can improve performance, these MC methods have low computing efficiency and scalability when dealing with large datasets.

In addition, nonnegativity and sparsity constraints, as well as regularization leading to various MF models, could be considered. Most MF models imply that numerous decomposition factors are nonnegative and extend some Nonnegative Matrix Factorization (NMF) [17] notions. NMF can learn a component-based representation rather than a global representation, as it permits only the additive combination of learned matrices (no subtraction). According to this approach, two low-rank nonnegative matrices are used to represent the objective matrix, and it is assumed that the original matrix rank is already known.

Single-layer matrix approximations are incapable of extracting multi-level features from complicated data sources. Recently, deep matrix factorization was proposed to cope with extracting several layers of features. The fundamental purpose of deep MF is to combine interpretability, as in conventional Matrix Factorizations, and the extraction of numerous hierarchical features, as permitted by multi-layer structures [18]. An approach termed Deep Matrix Factorization (DMF) for non-linear matrix completion is proposed in [19]. This model uses a non-linear latent variable model to complete the matrices. DMF is formulated as a deep-structure neural network in which the inputs are low-dimensional latent variables, and the outputs are partially seen variables. Also, [20] proposed a patch-based non-linear matrix completion technique. The predictive link between a matrix entry and its surrounding entries is learned by an end-to-end trainable model that leads to a non-linear MC solution in this technique, which employs a Convolutional Neural Network. Autoencoder-based Matrix Completion (AEMC) is an approach provided in [21]. Autoencoder parameters and missing data matrix entries are updated simultaneously in AEMC to minimize reconstruction errors. The hidden units of the AEMC Autoencoder are smaller than the visible units, and redundant variables can be compressed into fewer latent variables. It also means that a subset of the input variables may be used to reconstruct latent variables, which can then generate the rest of the input variables. In [22], *Mehrdad and Kahaei* provided a mix of linear and non-linear latent variables for the data matrix, and a regularized deep neural network-based technique for handling linear and non-linear relationships among data matrix elements.

Deep structure models have a fundamental challenge in that they are considerably overparameterized in practice: the number of learnable parameters is much higher than the number of training examples. The objective has numerous global minima in an overparameterized setting, all of which minimize the training error, but many do not generalize well. As a result, minimizing the training error is insufficient for different learning tasks. Adversarial training (AT) has recently been highlighted to cover this challenge and increase generalization. The robustness of adversarially-trained deep learning models is demonstrated in [23]. More recently, *Sinha et al.* [24] and *Farnia et al.* [25] have shown the improvement of AT in machine learning generalizations.

Also, *Szegedy et al.* [26] were the first to provide the phenomena of adversarial instances, who discovered that injecting small but purposefully generated perturbations to valid inputs might cause deep learning networks to make inaccurate 2D image classification choices. Many adversarial attacks in many fields have been developed using increasingly efficient and effective algorithms [27]. Adversarial perturbation is a type of noise that is purposefully designed to fool deep learning-based models while causing little perceptual distortion to the image. Many approaches have been proposed to mislead adversarial image classifiers. For example, a Prototype-Supervised Generative Adversarial Network (ProS-GAN) is proposed for flexible targeted hashing attacks [28,29].

Luo et al. [30] presented a novel Adversarial Nonnegative Matrix Factorization model to guarantee robustness against actual perturbations. The desired feature and weight matrices are learned while data is subjected to adversarial perturbations. Instead of focusing on the regular data points, their models focus on possible test adversaries outside the specified bounds. The suggested model is formulated as a bi-level optimization problem, solved using ADMM. An adversarially-trained variant of the NMF, a common latent dimensionality reduction approach, is also considered in [31]. They proposed that an attacker can manipulate the data matrix by adding an arbitrarily large matrix of the bounded norms. In addition, they provided methods based on adversarial training to optimize for dictionary and coefficient matrices with improved generalization abilities.

As part of this research, the Robust Matrix Completion (RMC) problem is investigated. The aim is to recover a low-rank matrix with a limited number of sparsely corrupted elements. Conventional models with the least square error function are excellent for zero-mean Gaussian noise measurement. However, many problems may not correspond with the condition above in different real-world applications. The standard learners with the least square error ℓ_2 are sensitive to outliers [32], which may significantly influence a square residue loss on the objective function. The robust learner with ℓ_1 norm is ideally performed in the Laplacian noise model [32]. When the noise distribution is unpredictable, elastic models include an elastic loss to utilize the advantages of both the ℓ_2 and ℓ_1 norms; therefore, an elastic model is much less sensitive to noises and outliers [33]. Recently, *Xiong and Kong* proposed an elastic NMF model that uses an elastic loss intercalated between Frobenius and $\ell_{2,1}$-norm to evaluate the factorization quality [34]. To estimate the low-rank matrix robustly, *Cherapanamjeri et al.* [35] presented a projected gradient descent-based technique that alternates between performing a projected gradient descent step and utilizing hard-thresholding to clear up a few of the corrupted entries. This technique solves RMC with a near-optimal amount of observations and corruption tolerance. *Klopp et al.* [36] addressed the problem that the seen entries are noisy perceptions of a sum of a low-rank matrix that they wanted to approximate and a second matrix with a sparse complementary structure such as element-wise sparsity or column-wise sparsity. *He et al.* [37] utilize correntropy as the loss function to enhance the robustness. They apply the half-quadratic method to overcome the computation of non-quadratic correntropy-based optimization.

This paper aims to design a more robust adversarial model for generating a wider range of attacks. Motivated by the generalization properties of adversarial training and robustness properties of different matrix norms, we proposed a new robust adversarial deep matrix completion algorithm called Elastic Adversarial Deep Nonnegative Matrix Factorization (EADNMF), which utilizes an adaptive combination of Frobenius and $\ell_{2,1}$ losses in its training process. This dynamic model, by emphasizing potential test adversaries beyond the pre-defined constraints, effectively prevents overfitting and allows for building a more accurate and reliable model with a limited number of observations. For this purpose, an attempt has been made to introduce an elastic matrix completion method, which can be insensitive to outliers and perturbed data and cover a mixture of multiple attacks. Mathematically, we explicitly cast EADNMF as a Majorization-Minimization optimization problem, and an efficient algorithm to solve the problem is provided. The main contributions of this paper are summarized as follows:

- The underlying model aims at learning a Deep Matrix Factorization to approximate the complex latent model based on unknown and diverse attributes of the original high-dimensional data. This model decomposes an incomplete matrix into multiple basis and coefficient matrices through the deep structure.
- In the basic model, we consider the robust adversarial matrix completion problem, where the goal is to recover a low-rank matrix by observing a small number of its elements. This method focuses on robustness against attacks to introduce the Adversarial Deep NMF (ADNMF) model with a Frobenius norm.
- One major difficulty of adversarial training is that it tends to overfit adversarial examples. To cover this deficiency, we introduce an elastic loss which leverages the advantage of both Frobenius and $\ell_{2,1}$ norms when attack distribution is uncertain. Hence, the EADNMF model boosts the robustness of adversarial training and improves the generalization performance.

The rest of this paper is organized as follows: first, the background of Nonnegative Matrix Factorization and Deep Nonnegative Matrix Factorization is introduced in Section 2. In Section 3, the proposed models and their numerical solutions are presented. In Section 4, the advantages of the proposed method are experimentally validated. Finally, the conclusion will be provided in Section 5.

2. Background

2.1. Notations

In this paper, scalars are shown by lowercase italic notations (i.e., i, j, n, etc.) and uppercase italic notations (i.e., A, B, M, etc.) while bold lowercase notations (i.e., $\boldsymbol{a}, \boldsymbol{x}$, etc.) and bold uppercase notations (i.e., $\boldsymbol{W}, \boldsymbol{H}$, etc.) show vectors and matrices, respectively. For any matrix $\boldsymbol{A}$, $\boldsymbol{a}^{(i)}$ means the i-th column of $\boldsymbol{A}$, $\boldsymbol{a}_i$ means the i-th row of $\boldsymbol{A}$, and A_{ij} denotes the (i,j)-element of $\boldsymbol{A}$. $\mathrm{Tr}(\boldsymbol{A})$ is the trace of $\boldsymbol{A}$, and the transposed matrix of $\boldsymbol{A}$ is shown by $\boldsymbol{A}^\top$. The *Frobenius* norm of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is introduced as $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2} = \sqrt{\mathrm{Tr}(\boldsymbol{A}^\top \boldsymbol{A})}$. The $\ell_{2,1}$-norm for $\boldsymbol{A}$ is defined as $\|\boldsymbol{A}\|_{2,1} = \sum_{j=1}^n \|\boldsymbol{a}^{(j)}\| = \sum_{j=1}^n \sqrt{\sum_{i=1}^m A_{ij}^2}$.

2.2. Nonnegative matrix factorization

Given a d dimensional vector $\boldsymbol{v}$ with nonnegative entries, whose n observations are denoted as $\boldsymbol{v}^{(j)}, j = 1, 2, \ldots, n$, let the input matrix be $\boldsymbol{V} = [\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(n)}] \in \mathbb{R}_{\geqslant 0}^{d\times n}$, NMF seeks to factorize $\boldsymbol{V}$ into nonnegative basis

$\boldsymbol{W} = [\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \ldots, \boldsymbol{W}^{(r)}] \in \mathbb{R}_{\geqslant 0}^{d\times r}$ and nonnegative $r \times n$ coefficient matrices $\boldsymbol{H} = [\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \ldots, \boldsymbol{h}^{(n)}] \in \mathbb{R}_{\geqslant 0}^{r\times n}$, such that $\boldsymbol{V} \approx \boldsymbol{WH}$ [17].

Given the observation $\boldsymbol{v}^{(i)} \in \mathbb{R}_{\geqslant 0}^{d}$ (ith data point), NMF decomposes it into the basis $\boldsymbol{W} \in \mathbb{R}_{\geqslant 0}^{d\times r}$ and the representation $\boldsymbol{h}^{(i)} \in \mathbb{R}_{\geqslant 0}^{r}$, i.e.,

$$\min_{\boldsymbol{W},\boldsymbol{h}\geqslant 0} \sum_i z(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}), \tag{1}$$

where $z(.)$ is a loss function.

It is obvious that $\boldsymbol{h}^{(i)}$ is the weight coefficient of the seen entries $\boldsymbol{v}^{(i)}$ on the columns of $\boldsymbol{W}$, the latent vectors of $\boldsymbol{V}$. Therefore, NMF factors each data into the linear or non-linear combination of the base vectors. Because of the precondition $r \ll min(d, n)$, the obtained base vectors are incomplete over the original space. In other words, this approach attempts to represent a high-dimensional pattern with far fewer bases. Therefore, the perfect approximation can be achieved successfully only if the intrinsic features are identified in $\boldsymbol{W}$. Numerous types of losses are commonly used in NMF, such as least square NMF:

$$z_2(\boldsymbol{V}, \boldsymbol{WH}) = \sum_i \|\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)}\|^2 = \|\boldsymbol{V} - \boldsymbol{WH}\|_F^2, \tag{2}$$

and $\ell_{2,1}$ NMF:

$$z_{2,1}(\boldsymbol{V}, \boldsymbol{WH}) = \sum_i \|\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)}\| = \|\boldsymbol{V} - \boldsymbol{WH}\|_{2,1}. \tag{3}$$

2.3. Deep nonnegative matrix factorization

Real-world data frequently have complex and diverse organizing patterns. As a result, it is quite likely that the mapping between the original data and the latent space contains rather complicated hierarchical and structural information with implicit lower-level hidden characteristics. Deep learning is widely recognized for its ability to close the gap between the lower-level abstraction and the higher-level abstraction of the original data [38].

We introduce the Deep Nonnegative Matrix Factorization problem, which factorizes an input matrix $\boldsymbol{V}$ into $p + 1$ factors,

$$\boldsymbol{V} \approx \boldsymbol{W}_1\boldsymbol{W}_2 \ldots \boldsymbol{W}_p\boldsymbol{H}_p \tag{4}$$

where $\boldsymbol{H}_p \in \mathbb{R}_+^{r_p\times n}$ is a low-dimensional representation, $\boldsymbol{W}_l \in \mathbb{R}_+^{r_{l-1}\times r_l}(1 \leqslant l \leqslant p)$ is the mapping matrix of the lth layer.

The formulation in (4) provides a hierarchy of p layers of implicit representations of the input matrix, which the following decompositions can obtain:

$$\begin{aligned} \boldsymbol{H}_{p-1} &\approx \boldsymbol{W}_p\boldsymbol{H}_p \\ &\vdots \\ \boldsymbol{H}_2 &\approx \boldsymbol{W}_3 \ldots \boldsymbol{W}_p\boldsymbol{H}_p \\ \boldsymbol{H}_1 &\approx \boldsymbol{W}_2 \ldots \boldsymbol{W}_p\boldsymbol{H}_p \end{aligned} \tag{5}$$

The objective function of the Deep NMF algorithm can be defined as

$$\min_{\boldsymbol{W}_l,\boldsymbol{H}_p} \|\boldsymbol{V} - \boldsymbol{W}_1 \ldots \boldsymbol{W}_p\boldsymbol{H}_p\|_F^2, \text{s.t.} \quad \boldsymbol{W}_l \geqslant 0, \boldsymbol{H}_p \geqslant 0, \forall l = 1, \ldots, p. \tag{6}$$

3. Proposed model

In this part, inspired by adversarial training and elastic loss, an efficient algorithm is developed to optimize a Deep Nonnegative Matrix Factorization with enhanced generalization abilities. There are four major factors for its success: (1) It leverages a deep low-rank structure of the complex data matrix to obtain a more abstract latent representation, which also provides a natural solution to missing entries recovery (Section 3.1). (2) An adversarial training model employs an attack approach as a regularization for training Deep NMF. The training process may be conducted on the recently proposed adversaries to robustify models against perturbations (Section 3.2). (3) It learns the latent variables of all the samples by utilizing an elastic loss function in the adversarial matrix completion to control training. As a result, it is much less insensitive to outliers and noise (Section 3.3). (4) It incorporates those above into one joint learning problem and applies an effective alternating minimization method for optimization (Section 3.4). The structure of the EADNMF model is shown in Fig. 1.

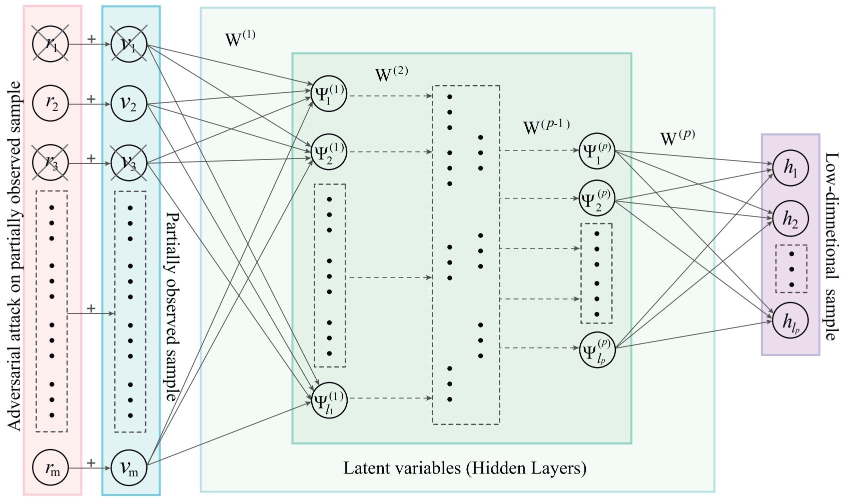


Fig. 1. The architecture of Elastic Adversarial Deep NMF (EADNMF).

3.1. Deep matrix completion by NMF

Given an incomplete matrix $\hat{\boldsymbol{V}} \in \mathbb{R}^{d\times n}$, matrix completion is to recover the missing elements of $\boldsymbol{V}$ by the seen elements. A common assumption for matrix completion is that the observed elements are sampled from a uniform random. Most of the classic matrix completion methods assume that $\boldsymbol{V}$ is of low-rank, i.e., $rank(\boldsymbol{V}) = r \ll min(d,n)$. Hence, the missing elements of $\boldsymbol{V}$ can be recovered by rank-minimization. Because direct minimizing rank is NP-hard, rank is usually approximated with nuclear-norm [7], truncated nuclear-norm [2], or other methods [39]. With the low-rank property, $\hat{\boldsymbol{V}}$ can be decomposed as

$$\min_{\boldsymbol{W}_l,\boldsymbol{H}_p} \|\Pi_\Omega(\boldsymbol{V} - \boldsymbol{W}_1 \dots \boldsymbol{W}_p\boldsymbol{H}_p)\|_F^2 = \|\boldsymbol{J} \odot (\boldsymbol{V} - \boldsymbol{W}_1 \dots \boldsymbol{W}_p\boldsymbol{H}_p)\|_F^2, \qquad (7)$$
$$\text{s.t.} \quad \boldsymbol{W}_l \geqslant 0, \boldsymbol{H}_p \geqslant 0, \forall l = 1,\dots,p.$$

Assume Ω be the set containing indices of the seen elements in $\boldsymbol{V}$ and $\boldsymbol{J} = [J_{ij}]$ be the indicator observation matrix with $J_{ij} = 1$ if $(i,j) \in \Omega$, and 0 otherwise.

3.2. Basic adversarial model

Most Nonnegative Matrix Factorization methods are non-convex optimization problems; therefore, they usually get stuck at the local minimum, especially when missing data and outliers are prevalent. Furthermore, increasing model capacity by adding depth increases training power, and greater training power is associated with a greater potential risk of overfitting. As a result, we want to address these problems by employing Adversarial Training since deep matrix factorizations are highly susceptible to adversarial attacks.

This section introduces the Adversarial Deep NMF (ADNMF) formulation. We suppose that an attacker exists, which adds an arbitrary attack $\boldsymbol{R} \in \mathbb{R}^{d\times n}$to the input matrix $\boldsymbol{V}$. The adversarial matrix $\boldsymbol{R}$ maximizes the least square loss (Frobenius norm) between $\boldsymbol{V}$ and the prediction $\boldsymbol{W}_1 \dots \boldsymbol{W}_p\boldsymbol{H}_p$. Because the adversary usually has limited strength, we suppose $\boldsymbol{R}$ belongs to a bounded set. As a result, ADNMF can be written as

$$\min_{\boldsymbol{W}_l,\boldsymbol{H}_p}\max_{\boldsymbol{R}\in\mathcal{R}} \|\boldsymbol{J} \odot (\boldsymbol{V} + \boldsymbol{R} - \boldsymbol{W}_1 \dots \boldsymbol{W}_p\boldsymbol{H}_p)\|_F^2, \text{s.t.} \quad \boldsymbol{W}_l \geqslant 0, \boldsymbol{H}_p \geqslant 0, \forall l = 1,\dots,p. \qquad (8)$$

where the defined bound constraint is

$$\mathcal{R} := \{\boldsymbol{R} : \|\boldsymbol{R}\|_F^2 \leqslant \epsilon, \boldsymbol{V} + \boldsymbol{R} \geqslant 0\} \qquad (9)$$

Here, constant $\epsilon > 0$ is the adversary's power; the larger ϵ leads to a strong adversary and vice versa. We suppose that $\boldsymbol{R}$ belongs to the multiple Frobenius-norm bounded matrices. $\boldsymbol{V} + \boldsymbol{R}$ is forced to be nonnegative so that for any fixed $\boldsymbol{R}$, the optimization in (8) corresponds to a DNMF problem with effective input matrix $\boldsymbol{V} + \boldsymbol{R}$. We note that a similar min–max function has recently appeared in the literature [40], but their goal was different: they only wanted to be robust to the β-divergence measure used and not perturbations to the input matrix.

The problem mentioned in (8) and (9) is tough to solve. Therefore, inspired by Lagrangian duality, we utilize relaxation [41]. The norm constraint $\|\boldsymbol{R}\|_F^2 \leqslant \epsilon$ in (9) is dualized with a Lagrange multiplier $\lambda > 0$; as a result, the basic cost function is

$$\min_{\boldsymbol{W}_l,\boldsymbol{H}_p}\max_{\boldsymbol{R}} \|\boldsymbol{J}\odot(\boldsymbol{V}+\boldsymbol{R}-\boldsymbol{W}_1\ldots\boldsymbol{W}_p\boldsymbol{H}_p)\|_F^2-\lambda\|\boldsymbol{R}\|_F^2, \quad \text{s.t.} \quad \boldsymbol{V}+\boldsymbol{R}\geqslant 0, \boldsymbol{W}_l\geqslant 0, \boldsymbol{H}_p\geqslant 0, \forall l=1,\ldots,p. \tag{10}$$

The inner optimization is a maximization; therefore, a negative regularization term is added to (10). The interpretation of λ is dual to that of ϵ; it demonstrates the power of the adversary. In fact, as $\lambda\to 0^+$, $\|\boldsymbol{R}\|_F^2$ will be unbounded, implying a very powerful adversary. In contrast, as $\lambda\to\infty$, the impact of the regularization term fades away, and $\|\boldsymbol{R}\|_F^2$ bends to zero, implying that there is no adversary.

The optimization in (10) is naturally divided into two sections, an inner maximization that optimize $\boldsymbol{R}$ and an outer minimization that optimize $\boldsymbol{W}_1,\ldots,\boldsymbol{W}_p,\boldsymbol{H}_p$. The inner maximization can be rewritten in the same way, as a minimization as

$$\boldsymbol{R}^* = \operatorname{argmin}_{\boldsymbol{R}} -\|\boldsymbol{J}\odot(\boldsymbol{V}+\boldsymbol{R}-\boldsymbol{W}_1\ldots\boldsymbol{W}_p\boldsymbol{H}_p)\|_F^2+\lambda\|\boldsymbol{R}\|_F^2, \text{s.t.} \quad \boldsymbol{V}+\boldsymbol{R}\geqslant 0. \tag{11}$$

In the next section, we show that the solution to this minimization is well-defined in a closed-form. The following objective function can be optimized after $\boldsymbol{R}^*$ has been obtained over $\boldsymbol{W}_1,\ldots,\boldsymbol{W}_p,\boldsymbol{H}_p\geqslant 0$.

$$\min_{\boldsymbol{W}_l,\boldsymbol{H}_p}\|\boldsymbol{J}\odot(\boldsymbol{V}+\boldsymbol{R}^*-\boldsymbol{W}_1\ldots\boldsymbol{W}_p\boldsymbol{H}_p)\|_F^2, \text{s.t.} \quad \boldsymbol{W}_l\geqslant 0, \boldsymbol{H}_p\geqslant 0, \forall l=1,\ldots,p. \tag{12}$$

Through the new input matrix $\boldsymbol{V}+\boldsymbol{R}^*$, we can utilize Majorization-Minimization (MM) [42] to obtain solutions for $\boldsymbol{W}_1,\ldots,\boldsymbol{W}_p,\boldsymbol{H}_p$. We should mention that while employing the least square loss (Frobenius norm), we have defined a model sensitive to perturbation. Nevertheless, there are superior substitutes to Majorization-Minimization.

3.3. Elastic adversarial model

Adversarial training proved the best method to defend against white-box attacks [43]. The model can perform better if we use strong defenses in matrix completion. In this way, the elastic model has confused the attacker by adapting its loss function. In this section, we will introduce a loss function that is insensitive to attacks and outliers.

The various loss functions have resulted in the NMF extends, such as $\ell_{2,1}$ -NMF [44]. Inspired by the idea of elastic net [33,45], a robust loss function called **elastic loss** is introduced, to fit the matrix factorization [34]. The elastic loss function is introduced as:

$$z_{\boldsymbol{el}}(\boldsymbol{V},\boldsymbol{WH}) = \sum_i\|\boldsymbol{v}^{(i)}-\boldsymbol{Wh}^{(i)}\|^2+\beta_i\sum_i\|\boldsymbol{v}^{(i)}-\boldsymbol{Wh}^{(i)}\|, \tag{13}$$

where β_i is the loss parameter. The matrix-wise cost function can be rewritten as

$$z_{\boldsymbol{el}}(\boldsymbol{V},\boldsymbol{WH}) = \|\boldsymbol{V}-\boldsymbol{WH}\|_F^2+\|diag(\beta)(\boldsymbol{V}-\boldsymbol{WH})\|_{2,1}. \tag{14}$$

It is worth noting that $z_{\boldsymbol{el}}(.)$ is nonnegative and convex, which is generally ideal for a cost function. It is a suitable candidate for the cost function and might be useful in some tasks such a as classification, clustering, and collaborative filtering. Albeit, $z_{\boldsymbol{el}}(.)$ is non-smooth at some points as standard ℓ_1 loss. Basically, (13) can be considered a trade-off between ℓ_2 loss and ℓ_1 loss, making it robust to outliers and noises. In the case that all β_i values are fixed irrespective of the scales of residues, we call (13) a hard-elastic loss. A closed-form of relaxed loss, namely soft-elastic loss, is defined as $h(.)$:

$$h_{\boldsymbol{el}}(\boldsymbol{V},\boldsymbol{WH}) = \underbrace{\sum_i\frac{\delta\|\boldsymbol{v}^{(i)}-\boldsymbol{Wh}^{(i)}\|^2}{\delta+\|\boldsymbol{v}^{(i)}-\boldsymbol{Wh}^{(i)}\|}}_{\ell_2\text{pseudo-loss}}+\underbrace{\sum_i\frac{\|\boldsymbol{v}^{(i)}-\boldsymbol{Wh}^{(i)}\|^2}{\delta+\|\boldsymbol{v}^{(i)}-\boldsymbol{Wh}^{(i)}\|}}_{\ell_1\text{pseudo-loss}} = \|\boldsymbol{V}-\boldsymbol{WH}\|_{\boldsymbol{el}}, \tag{15}$$

including ℓ_2 pseudo-loss and ℓ_1 pseudo-loss based on the scale parameter δ. Let ϵ_i denote the residue of the i-th instance, i.e., $\epsilon_i=\|\boldsymbol{v}^{(i)}-\boldsymbol{Wh}^{(i)}\|$. If ϵ_i is large, $\boldsymbol{v}^{(i)}$ might be noisy. In this case, the loss function (15) prefers to emphasize more on ℓ_1 loss. On the other hand, if ϵ_i is small, $\boldsymbol{v}^{(i)}$ is clear. In this case, the loss function (15) prefers to emphasize more on ℓ_2 loss. For more details, the reader is referred to A.

Given all the above preparations, we are now prepared to exhibit the Adversarial Deep elastic NMF model. The final objective function for EADNMF is:

$$\min_{\boldsymbol{W}_l,\boldsymbol{H}_p}\max_{\boldsymbol{R}} \|\boldsymbol{J}\odot(\boldsymbol{V}+\boldsymbol{R}-\boldsymbol{W}_1\ldots\boldsymbol{W}_p\boldsymbol{H}_p)\|_{\boldsymbol{el}}-\lambda\|\boldsymbol{R}\|_F^2, \quad \text{s.t.} \quad \boldsymbol{V}+\boldsymbol{R}\geqslant 0, \boldsymbol{W}_i\geqslant 0, \boldsymbol{H}_p\geqslant 0, \forall i=1,\ldots,p. \tag{16}$$

From the defense point of view, it can be said that when an instance is attacked, the learner can withstand abnormal attacks in addition to learning normal examples due to the flexibility of its cost function. So, the attacker cannot easily fool the learner. From another perspective, it can be claimed that the attacker, by knowing the elasticity of the model, tries to attack the features of each instance more intelligently so that the learner does not recognize the attack easily. Therefore, the proposed adversarial model will be more efficient than the previous methods in defense and attack aspects.

3.4. Numerical solution

Problem 16 can be addressed using alternating minimization (Algorithm 1), which allows us to update the variables iteratively until we reach a satisfactory solution. We use gradient descent to update one of the variables and fix the others in each iteration. The entire optimization problem is then broken down into smaller, easier-to-solve subproblems.

3.4.1. Optimizing attacker

In this part, we specify how to solve $\boldsymbol{R}^*$ in (11). Here $\boldsymbol{W}_l$ and $\boldsymbol{H}_l$ matrices are considered fixed, and their product is shown as $\bar{\boldsymbol{V}} = \boldsymbol{W}_1 \dots \boldsymbol{W}_p \boldsymbol{H}_p$. Thus, we can denote the objective as

$$\begin{aligned} g(\boldsymbol{R}) &:= -\|\boldsymbol{J} \odot (\boldsymbol{V} + \boldsymbol{R} - \bar{\boldsymbol{V}})\|_{\boldsymbol{el}} + \lambda \|\boldsymbol{R}\|_F^2 \\ &= \mathrm{Tr}(-(\boldsymbol{J} \odot \boldsymbol{R})\boldsymbol{D}(\boldsymbol{J} \odot \boldsymbol{R})^\top - 2(\boldsymbol{J} \odot \boldsymbol{V})\boldsymbol{D}(\boldsymbol{J} \odot \boldsymbol{R})^\top + 2(\boldsymbol{J} \odot \boldsymbol{R})\boldsymbol{D}\left(\boldsymbol{J} \odot \bar{\boldsymbol{V}}\right)^\top + \lambda \boldsymbol{R}\boldsymbol{R}^\top) \end{aligned} \tag{17}$$

where diagonal matrix $\boldsymbol{D}$ with the diagonal elements is defined [34] as

$$D_{ii} = (1 + \delta) \frac{2\delta + \|\boldsymbol{j}^{(i)} \odot (\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)})\|}{2(\delta + \|\boldsymbol{j}^{(i)} \odot (\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)})\|)^2}, \tag{18}$$

It should be noted that in the case of the proposed model without elastic (i.e. ADNMF), it is enough to set the $\boldsymbol{D} = \boldsymbol{I}$ in all formulas.

The optimal and unrealistic value for R_{ij} is ∞, which demonstrates an unbounded adversary power. If $\overline{V}_{ij} - V_{ij} > 0$, then the solution is $R^*_{ij} = -V_{ij}$; if $\overline{V}_{ij} = V_{ij}$, regardless of the value of R_{ij}, the function is 0; finally, if $\overline{V}_{ij} - V_{ij} < 0$, the solution is $R^*_{ij} = \infty$. According to these conditions, the convex objective function is easy to represent by a direct differentiation

$$\frac{\partial g}{\partial \boldsymbol{R}} = -2(\boldsymbol{J} \odot \boldsymbol{R})\boldsymbol{D} - 2(\boldsymbol{J} \odot \boldsymbol{V})\boldsymbol{D} + 2(\boldsymbol{J} \odot \bar{\boldsymbol{V}})\boldsymbol{D} + 2\lambda \boldsymbol{R} = 0 \tag{19}$$

and

$$\lambda \boldsymbol{R} - (\boldsymbol{J} \odot \boldsymbol{R})\boldsymbol{D} = \boldsymbol{J} \odot (\boldsymbol{V} - \bar{\boldsymbol{V}})\boldsymbol{D} \tag{20}$$

Under $\boldsymbol{V}_{ij} + \boldsymbol{R}_{ij} \geqslant 0$ condition, optimal solution for (17) is

$$\boldsymbol{R} = \max\left\{\boldsymbol{J} \odot (\boldsymbol{V} - \bar{\boldsymbol{V}}) \frac{\boldsymbol{D}}{\lambda \boldsymbol{I} - \boldsymbol{D}}, -(\boldsymbol{J} \odot \boldsymbol{V})\right\} \tag{21}$$

where we use an element-wise max operator.

3.4.2. Optimizing learner

To accelerate the approximation of the variables in the proposed method, we pre-train each layer by initial values for matrices $\boldsymbol{W}_l$ and $\boldsymbol{H}_l$. This pre-training can significantly decrease the learning time of the proposed method. *Hinton et al.* [46] have proved pre-training effectiveness on deep neural networks. To apply the pre-training, we first factor the incomplete matrix $\hat{\boldsymbol{V}} \approx \boldsymbol{W}_1 \boldsymbol{H}_1$ by minimizing $\|\boldsymbol{J} \odot (\boldsymbol{V} - \boldsymbol{W}_1 \boldsymbol{H}_1)\|_F^2$ where $\boldsymbol{W}_1 \in \mathbb{R}_+^{d \times r_1}$ and $\boldsymbol{H}_1 \in \mathbb{R}_+^{r_1 \times n}$. Then, we factor the matrix $\boldsymbol{H}_1$ as $\boldsymbol{H}_1 \approx \boldsymbol{W}_2 \boldsymbol{H}_2$ by minimizing $\|\boldsymbol{H}_1 - \boldsymbol{W}_2 \boldsymbol{H}_2\|_F^2$ where $\boldsymbol{W}_2 \in \mathbb{R}_+^{r_1 \times r_2}$ and $\boldsymbol{H}_2 \in \mathbb{R}_+^{r_2 \times n}$. This process will be repeated until all of the layers have been initialized. Finally, each layer is fine-tuned by alternating optimization of the proposed model in Eq. (16). We provide the update rules in the following.

Update rule for the basis matrix $\boldsymbol{W}_l$ $(1 \leqslant l \leqslant p)$ By fixing all the matrices (except $\boldsymbol{W}_l$), the objective function in Eq. (16) is:

$$\begin{aligned} &\min_{\boldsymbol{W}_l} \|\boldsymbol{J} \odot (\boldsymbol{V}_R - \Psi_{l-1} \boldsymbol{W}_l \Phi_{l+1} \boldsymbol{H}_p)\|_{\boldsymbol{el}} - Tr(\Theta_l \boldsymbol{W}_l^\top) \\ &\quad = Tr[\boldsymbol{J} \odot (\Psi_{l-1} \boldsymbol{W}_l \Phi_{l+1} \boldsymbol{H}_p)\boldsymbol{D}\left(\left(\boldsymbol{H}_p^\top \Phi_{l+1}^\top \boldsymbol{W}_l^\top \Psi_{l-1}^\top\right) \odot \boldsymbol{J}^\top\right) - 2(\boldsymbol{J} \odot \boldsymbol{V}_R)\boldsymbol{D}\left(\left(\boldsymbol{H}_p^\top \Phi_{l+1}^\top \boldsymbol{W}_l^\top \Psi_{l-1}^\top\right) \odot \boldsymbol{J}^\top\right)] - Tr(\Theta_l \boldsymbol{W}_l^\top) \end{aligned} \tag{22}$$

where $\boldsymbol{V}_R = \boldsymbol{V} + \boldsymbol{R}, \Psi_{l-1} = \boldsymbol{W}_1 \dots \boldsymbol{W}_{l-1}$, and $\Phi_{l+1} = \boldsymbol{W}_{l+1} \dots \boldsymbol{W}_p$. When $l = 1$, we set $\Psi_0 = \boldsymbol{I}$. Similarly, where $l = p$, we consider $\Phi_{p+1} = \boldsymbol{I}$.

To solve Eq. (22), we define a Lagrangian multiplier Θ_l to force the nonnegativity on $\boldsymbol{W}_l$. By setting the partial derivative of (22) with respect to $\boldsymbol{W}_l$ to $\boldsymbol{0}$, we obtain:

$$\Theta_l = 2\Psi_{l-1}^\top (\boldsymbol{J} \odot (\Psi_{l-1} \boldsymbol{W}_l \Phi_{l+1} \boldsymbol{H}_p))\boldsymbol{D}\boldsymbol{H}_p^\top \Phi_{l+1}^\top - 2\Psi_{l-1}^\top (\boldsymbol{J} \odot \boldsymbol{V}_R)\boldsymbol{D}\boldsymbol{H}_p^\top \Phi_{l+1}^\top \tag{23}$$

With respect to the complementary slackness condition of the Karush–Kuhn–Tucker, we have:

$$\Theta_l \odot \boldsymbol{W}_l = \boldsymbol{0}, \tag{24}$$

where $\odot$ shows the *Hadamard* product. This equation is the fixed point that the solution must guarantee convergence. By solving (24), we have the following update rule:

$$\boldsymbol{W}_l \leftarrow \boldsymbol{W}_l \odot \frac{\Psi_{l-1}^{\top}(\boldsymbol{J} \odot \boldsymbol{V}_R)\boldsymbol{D}\boldsymbol{H}_p^{\top}\Phi_{l+1}^{\top}}{\Psi_{l-1}^{\top}(\boldsymbol{J} \odot (\Psi_{l-1}\boldsymbol{W}_l\Phi_{l+1}\boldsymbol{H}_p))\boldsymbol{D}\boldsymbol{H}_p^{\top}\Phi_{l+1}^{\top}} \tag{25}$$

Update rule for the latent matrix $\boldsymbol{H}_p$ By fixing all the matrices except for $\boldsymbol{H}_p$, the objective function in (16) is:

$$\begin{aligned}&\min_{\boldsymbol{H}_p} \|\boldsymbol{J} \odot (\boldsymbol{V}_R - \Psi_p\boldsymbol{H}_p)\|_{\boldsymbol{el}} - \mathrm{Tr}\left(\Xi_p\boldsymbol{H}_p^{\top}\right) \qquad (26)\\ &= \mathrm{Tr}[\boldsymbol{J} \odot (\Psi_p\boldsymbol{H}_p)\boldsymbol{D}\left(\left(\boldsymbol{H}_p^{\top}\Psi_p^{\top}\right) \odot \boldsymbol{J}^{\top}\right) - 2(\boldsymbol{J} \odot \boldsymbol{V}_R)\boldsymbol{D}\left(\left(\boldsymbol{H}_p^{\top}\Psi_p^{\top}\right) \odot \boldsymbol{J}^{\top}\right)] - Tr\left(\Xi_p\boldsymbol{H}_p^{\top}\right)\end{aligned}$$

The update rule for $\boldsymbol{H}_p$ is obtained following like derivation of the update rule for $\boldsymbol{W}^{(i)}$,

$$\boldsymbol{H}_p \leftarrow \boldsymbol{H}_p \odot \frac{\Psi_p^{\top}(\boldsymbol{J} \odot \boldsymbol{V}_R)\boldsymbol{D}}{\Psi_p^{\top}(\boldsymbol{J} \odot (\Psi_p\boldsymbol{H}_p))\boldsymbol{D}} \tag{27}$$

Update rule for the latent matrix $\boldsymbol{H}_l$ $(1 \leqslant l < p)$ The updating of $\boldsymbol{H}_l$ is optional, because it does not considerably affect the objective function in (16). However, extracting the latent features in each middle layer can lead to more accurate representation in further complex data structures. To obtain an optimal solution for $\boldsymbol{H}_l$, we seek to optimize the following function:

$$\min_{\boldsymbol{H}_l} \|\boldsymbol{J} \odot (\boldsymbol{V}_R - \Psi_l\boldsymbol{H}_l)\|_{\boldsymbol{el}}, \text{s.t.} \quad \boldsymbol{H}_l \geqslant 0, \tag{28}$$

Similar to $\boldsymbol{H}_p, \boldsymbol{H}_l$ can be updated by

$$\boldsymbol{H}_l \leftarrow \boldsymbol{H}_l \odot \frac{\Psi_l^{\top}(\boldsymbol{J} \odot \boldsymbol{V}_R)\boldsymbol{D}}{\Psi_l^{\top}(\boldsymbol{J} \odot (\Psi_l\boldsymbol{H}_l))\boldsymbol{D}} \tag{29}$$

The optimization process of EADNMF is provided in Algorithm 1, where the *ShallowNMF* function does the pre-training.

Algorithm 1: Elastic Adversarial Deep NMF (EADNMF)

```
Input: Data matrix $V$, Indicator Matrix $J$, layers size $r$, Regularization parameter $\lambda$, Scale parameter $\delta$;
Output: $W_l$ $(1 \leqslant l < p), H_l$ $(1 \leqslant l < p)$, and the reconstructed matrix $\bar{V} = W_1 \ldots W_p H_p$;
1: ▷ Pre-training process:
2: $W_1, H_1 \leftarrow$ShallowNMF$(J, V, r_1)$;
3: for $l = 2$ to $p$ do
4:   $W_l, H_l \leftarrow$ ShallowNMF$(H_{l-1}, r_l)$;
5: end for
6: ▷ Fine-tuning process:
7: while outer convergence (31) not reached do
8:   Update attack $R$ according to (21)
9:   while inner convergence (30) not reached do
10:      if cost is Frobenius then
11:        $D = I$        //ADNMF model
12:      els if cost is Elastic then
13:        Update diagonal matrix $D$ according to (18)
14:      end if
15:      for $l = 1$ to $p$ do
16:        $\Psi_{l-1} \leftarrow \prod_{\tau=1}^{l-1} W_\tau (\Psi_0 \leftarrow \mathbf{I})$;
17:        $\Phi_{l+1} \leftarrow \prod_{\tau=l+1}^{p} W_\tau (\Phi_{p+1} \leftarrow \mathbf{I})$;
18:        Update $W_l$ according to (25);
19:        $\Psi_l \leftarrow \Psi_{l-1} W_l$;
20:        Update $H_l$ according to (29) $(l < p$, optional) or according to (27) $(i = p)$;
21:      end for
22:   end while
23: end while
24: return $W_l, H_l, \forall l = 1, 2, \ldots, p$;
```

3.4.3. Stopping criteria

The optimization of $\boldsymbol{W}_l, \boldsymbol{H}_l$ is achieved by iterative minimization. In this part, we bring up the used criterion to terminate this inner minimization before updating a new $\boldsymbol{R}$. We look at the relative error of subsequent iterates and stop when it goes below a specified threshold ϵ_{in}, i.e., if $\boldsymbol{W}_l, \boldsymbol{H}_l$ shows the iterate of $\boldsymbol{W}_l^{(o,i)}, \boldsymbol{H}_l^{(o,i)}$ at the oth outer iteration and ith inner iteration and $\bar{\boldsymbol{V}}^{(o,i)} := \boldsymbol{W}_1^{(o,i)} \dots \boldsymbol{W}_p^{(o,i)} \boldsymbol{H}_p^{(o,i)}$. The inner minimization will be terminated once the inner iteration step i reaches

$$\left\| \frac{\bar{\boldsymbol{V}}^{(o,i+1)} + \bar{\boldsymbol{V}}^{(o,i)}}{\bar{\boldsymbol{V}}^{(o,i)}} \right\|_F < \epsilon_{in} \tag{30}$$

The entire algorithm is terminated once the outer iteration step o reaches

$$\left\| \frac{\bar{\boldsymbol{V}}^{(o+1,i)} + \bar{\boldsymbol{V}}^{(o,i)}}{\bar{\boldsymbol{V}}^{(o,i)}} \right\|_F < \epsilon_{out} \tag{31}$$

where i is the step of the inner iteration. The algorithm finishes the inner and outer iterations after reaching the predetermined iteration numbers, *maxIn*, and *maxOut* are reached.

3.5. Complexity analysis

The proposed model has two stages, a pre-training phase and a fine-tuning phase, as shown in Algorithm 1. Therefore, we analyze the complexity of the two phases separately. To simplify the analysis, we denote r as the maximal layer size among all layers, n is the number of instances, d is the number of original feature dimensions, and p is the number of layers. The pre-training phase, which is the less time-consuming part, includes the *ShallowNMF* algorithm. For this model, the computational complexity of each layer is $O(dnr)$. Therefore, the computational complexity of the p layers is $O(p(dnr))$. If the shallow model achieves convergence of t_p iterations, the overall computational complexity of the pre-training phase is $T_{pretrain} = O(t_p p(dnr))$. Similarly, in the fine-tuning phase, the computational complexity for the learner is $O(t_{out} t_{in} p(dnr))$, and the computational complexity for the attacker is $O(t_{out}(dn))$, where t_{in} and t_{out} are the numbers of iterations needed for the learner and attacker, respectively. Therefore, the total computational complexity of its fine-tuning phase is $T_{finetune} = O(t_{out}(t_{in} p(dnr) + dn))$. To sum up, the overall computational complexity of EADNMF is $T_{total} = T_{pretrain} + T_{finetune} = O(t_p p(dnr)) + O(t_{out}(t_{in} p(dnr) + dn))$.

4. Experimental

In this section, the proposed methods ADNMF (with ℓ_2 loss) and EADNMF (with elastic loss) will be compared with basic NMF and four state-of-the-art methods of matrix completion in the tasks of toy matrix completion, single-image inpainting, and multiple image inpainting. The methods include the Nonnegative Matrix Factorization (NMF) method solved by Multiplicative Update Rules, Deep Matrix Factorization [19] (DMF) based method solved by iRprop+ [47], Adversarial Nonnegative Matrix Factorization [30] (ANMF) method solved by Alternating Direction Method of Multipliers, Adversarially-Trained Nonnegative Matrix Factorization [31] (AT-NMF) method solved by Multiplicative Update Rules, and robust matrix factorization via half-quadratic optimization [37] (HQASD). The parameters of all compared methods are carefully adjusted to present their best performances.

4.1. Evaluation measure

The Root Mean Square Error (RMSE) evaluates the matrix completion performance. Let $\bar{\Omega} \subset \{1, \dots, d\} \times \{1, \dots, n\}$ be the set of miss entries of $\boldsymbol{V}$, and $\alpha = |\bar{\Omega}|/(d \times n)$ be percentage of miss entries. The prediction of $\{V_{i,j} : (i,j) \in \bar{\Omega}\}$ is $\{\overline{V}_{i,j} : (i,j) \in \bar{\Omega}\}$ where $\overline{V}_{i,j} = [\boldsymbol{W}_1 \dots \boldsymbol{W}_p \boldsymbol{H}_p]_{i,j}$, and the evaluation measure is

$$RMSE = \sqrt{\frac{1}{|\bar{\Omega}|} \sum_{(i,j) \in \Omega} \left(V_{i,j} - \overline{V}_{i,j}\right)^2} \tag{32}$$

Table 1
Characteristics of the Synthetic and real-world datasets.

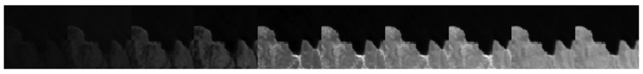
Dataset	#instance	#feature	Type
Synthetic	100	50	Synthetic
CBCL	2429	361	Facial Images
Moffet	165	2500	hyperspectral Images
Madonna	160	2500	hyperspectral Images
MNIST	1000	784	Handwriting Digit Images
Fashion-MNIST	1000	784	Clothing Images
MS-COCO-2017	5000	4096	Object Images

The smaller RMSE indicates a better prediction ability. We run each method with 10 independent initializations (described in Section 3). The means and standard deviations of the 10 runs are reported in Tables 2–8.

4.2. Datasets

We conduct numerical experiments on one synthetic and six real-world datasets to demonstrate the prediction and generalization abilities of ADNMF and EADNMF versus the other mentioned methods. The synthetic data is generated according to the described model in [31]. As the second dataset, we report *CBCL* dataset experiments [17]. This collection comprises 2429 face images, each with $d = 19 \times 19 = 361$ pixels; therefore, $V \in \mathbb{R}^{361\times 2429}$. The third and fourth datasets are the Moffet [5] and Madonna [48] hyperspectral image datasets. These datasets include $n = 165$ and $n = 160$ spectral bands, respectively. The image size (number of pixels) is 50×50; thus $d = 2500$. The *MNIST* digits handwriting is another famous dataset. This dataset consists of $N = 70000$ sample images and each having $28 \times 28 = 784$ pixels, included from '0' to '9' digit numbers. We sampled 1000 random images from this dataset. Thus, $V \in \mathbb{R}^{784\times 1000}$. The *Fashion-MNIST* [49] is a new complex dataset. This dataset consists of $N = 70000$ sample images; each example is a 28×28 grayscale image, and similar to *MNIST*, we

(a) 19×19 Samples of face *CBCL* dataset


(b) 50×50 Samples of hyperspectral *Moffet* dataset

(c) 50×50 Samples of hyperspectral *Madonna* dataset

(d) 28×28 Samples of handwriting *MNIST* dataset

(e) 28×28 Samples of clothes *Fashion-MNIST* dataset

(f) 64×64 Samples of *MS-COCO-2017* dataset

Fig. 2. six Grayscale images for multiple-image inpainting problem.

Table 2
Completion results on *Synthetic* dataset: α shows missing rates, and mean and std demonstrate the average and standard deviations over 10 independent runs, respectively. The best result is highlighted in **bold**, while the underline indicates the second-best.

α		NMF	ANMF	AT-NMF	HQASD	DMF	ADNMF	EADNMF
0.3	mean	5.37	6.78	5.41	0.75	0.57	**0.42**	**0.42**
	std	0.02	0.17	0.12	0.02	0.13	0.04	0.04
0.4	mean	5.62	6.92	5.54	0.76	0.59	**0.50**	0.50
	std	0.03	0.17	0.08	0.02	0.09	0.02	0.03
0.5	mean	6.41	7.44	6.27	1.07	0.76	0.69	**0.69**
	std	0.01	0.09	0.11	0.02	0.10	0.04	0.02
0.6	mean	6.74	7.61	6.47	1.22	1.01	**0.86**	**0.86**
	std	0.02	0.09	0.07	0.03	0.13	0.05	0.05
0.7	mean	7.3	7.99	7.02	1.75	1.59	**1.17**	1.18
	std	0.01	0.06	0.04	0.03	0.21	0.05	0.05
0.8	mean	7.87	8.30	7.69	3.35	2.77	1.54	**1.52**
	std	0.01	0.06	0.04	0.03	0.37	0.07	0.08
0.9	mean	8.45	8.58	8.44	5.04	3.83	2.08	**2.07**
	std	0.01	0.06	0.02	0.03	0.34	0.04	0.04

used 1000 random images of this dataset. Thus, $\boldsymbol{V} \in \mathbb{R}^{784\times1000}$. Finally, the *MS-COCO-2017* [50] is large-scale object detection, segmentation, and captioning dataset. We used 5000 validation images; each example is a 64×64 image. A summary of datasets is listed in Table 1, and sample images for each dataset are illustrated in Fig. 2.

Table 3
Completion results on *CBCL* face dataset: α shows missing rates, and mean and std demonstrate the average and standard deviations over 10 independent runs, respectively. The best result is highlighted in **bold**, while the <u>underline</u> indicates the second-best.

α		NMF	ANMF	AT-NMF	HQASD	DMF	ADNMF	EADNMF
0.3	mean	0.2040	0.1890	0.1760	<u>0.0806</u>	0.1113	0.0834	**0.0738**
	std	0.0010	0.0000	0.0010	0.0002	0.0002	0.0003	0.0003
0.4	mean	0.2320	0.2170	0.1920	<u>0.0845</u>	0.1120	0.0864	**0.0775**
	std	0.0010	0.0000	0.0010	0.0002	0.0003	0.0003	0.0003
0.5	mean	0.2530	0.2410	0.2090	<u>0.0895</u>	0.1128	0.0915	**0.0824**
	std	0.0010	0.0000	0.0010	0.0015	0.0002	0.0006	0.0004
0.6	mean	0.2720	0.2620	0.2270	0.1000	0.1140	<u>0.0984</u>	**0.0897**
	std	0.0000	0.0000	0.0000	0.0002	0.0003	0.0005	0.0003
0.7	mean	0.2910	0.2850	0.2480	0.1373	0.1165	<u>0.1106</u>	**0.0980**
	std	0.0000	0.0000	0.0010	0.0020	0.0002	0.0008	0.0003
0.8	mean	0.3090	0.3060	0.2770	0.2217	<u>0.1212</u>	0.1297	**0.1089**
	std	0.0000	0.0000	0.0010	0.0010	0.0003	0.0008	0.0003
0.9	mean	0.3280	0.3260	0.3140	0.2959	<u>0.1391</u>	0.1545	**0.1328**
	std	0.0000	0.0000	0.0000	0.0016	0.0007	0.0005	0.0003

Table 4
Completion results on *Moffet* hyperspectral dataset: α shows missing rates, and mean and std demonstrate the average and standard deviations over 10 independent runs, respectively. The best result is highlighted in **bold**, while the <u>underline</u> indicates the second-best.

α		NMF	ANMF	AT-NMF	HQASD	DMF	ADNMF	EADNMF
0.3	mean	0.0560	0.0500	0.0410	0.0222	0.0065	<u>0.0049</u>	**0.0035**
	std	0.0000	0.0000	0.0010	0.0009	0.0001	0.0001	0.0002
0.4	mean	0.0720	0.0670	0.0520	0.0264	0.0068	<u>0.0050</u>	**0.0036**
	std	0.0000	0.0000	0.0010	0.0008	0.0003	0.0002	0.0002
0.5	mean	0.0880	0.0840	0.0640	0.0350	0.0070	<u>0.0050</u>	**0.0036**
	std	0.0000	0.0000	0.0000	0.0006	0.0002	0.0002	0.0002
0.6	mean	0.1030	0.1000	0.0790	0.0485	0.0077	<u>0.0051</u>	**0.0037**
	std	0.0000	0.0000	0.0000	0.0005	0.0004	0.0001	0.0002
0.7	mean	0.1200	0.1170	0.0970	0.0731	0.0087	<u>0.0052</u>	**0.0038**
	std	0.0000	0.0000	0.0000	0.0006	0.0005	0.0001	0.0002
0.8	mean	0.1360	0.1340	0.1170	0.1032	0.0256	<u>0.0065</u>	**0.0044**
	std	0.0000	0.0000	0.0000	0.0003	0.0003	0.0002	0.0002
0.9	mean	0.1510	0.1500	0.1400	0.1271	0.0269	<u>0.0085</u>	**0.0057**
	std	0.0000	0.0000	0.0000	0.0003	0.0000	0.0004	0.0004

Table 5
Completion results on *Madonna* hyperspectral dataset: α shows missing rates, and mean and std demonstrate the average and standard deviations over 10 independent runs, respectively. The best result is highlighted in **bold**, while the <u>underline</u> indicates the second-best.

α		NMF	ANMF	AT-NMF	HQASD	DMF	ADNMF	EADNMF
0.3	mean	0.1200	0.1060	0.1030	0.0672	0.0113	<u>0.0099</u>	**0.0080**
	std	0.0000	0.0000	0.0060	0.0006	0.0020	0.0001	0.0002
0.4	mean	0.1470	0.1390	0.1170	0.0743	0.0146	<u>0.0100</u>	**0.0081**
	std	0.0000	0.0000	0.0040	0.0011	0.0020	0.0000	0.0002
0.5	mean	0.1750	0.1700	0.1310	0.0810	0.0148	<u>0.0103</u>	**0.0083**
	std	0.0000	0.0000	0.0060	0.0007	0.0022	0.0000	0.0002
0.6	mean	0.2040	0.2010	0.1580	0.0909	0.0159	<u>0.0103</u>	**0.0084**
	std	0.0000	0.0000	0.0030	0.0010	0.0016	0.0000	0.0002
0.7	mean	0.2320	0.2290	0.1850	0.1019	0.0167	<u>0.0105</u>	**0.0086**
	std	0.0000	0.0000	0.0050	0.0003	0.0001	0.0000	0.0002
0.8	mean	0.2610	0.2600	0.2270	0.1205	0.0174	<u>0.0105</u>	**0.0094**
	std	0.0000	0.0000	0.0020	0.0007	0.0003	0.0000	0.0003
0.9	mean	0.2900	0.2890	0.2710	0.1527	0.0200	<u>0.0112</u>	**0.0110**
	std	0.0000	0.0000	0.0000	0.0016	0.0001	0.0000	0.0001

4.3. Settings

All of the experiments involve predicting missed entries of $\boldsymbol{V}$. The fraction of missed entries is in the range of (0, 1), with values ranging from $\{0.3, 0.4, \ldots, 0.8, 0.9\}$. For example, when $\alpha = 0.9$, it means that 90% of entries of $\boldsymbol{V}$ are randomly

Table 6
Completion results on *MNIST* handwriting dataset: α shows missing rates, and mean and std demonstrate the average and standard deviations over 10 independent runs, respectively. The best result is highlighted in **bold**, while the <u>underline</u> indicates the second-best.

α		NMF	ANMF	AT-NMF	HQASD	DMF	ADNMF	EADNMF
0.3	mean	0.1989	0.1846	0.1825	<u>0.1574</u>	0.1911	0.1867	**0.1374**
	std	0.0008	0.000	0.0008	0.0005	0.0001	0.0006	0.0005
0.4	mean	0.2165	0.2024	0.1948	<u>0.1601</u>	0.1915	0.1873	<u>0.1393</u>
	std	0.0005	0.000	0.0012	0.0004	0.0002	0.0007	0.0006
0.5	mean	0.2356	0.2224	0.2154	<u>0.1634</u>	0.1929	0.1895	**0.1441**
	std	0.0005	0.000	0.0008	0.0005	0.0001	0.0004	0.0020
0.6	mean	0.2559	0.2435	0.2371	<u>0.1738</u>	0.1955	0.1927	**0.1536**
	std	0.0004	0.000	0.0008	0.0005	0.0002	0.0008	0.0005
0.7	mean	0.2762	0.2648	0.2586	0.2031	<u>0.1996</u>	0.1997	**0.1680**
	std	0.0005	0.000	0.0006	0.0012	0.0003	0.0006	0.0030
0.8	mean	0.2946	0.2857	0.2823	0.2672	<u>0.2093</u>	0.2241	**0.2000**
	std	0.0004	0.000	0.0005	0.0015	0.0003	0.0029	0.0010
0.9	mean	0.3101	0.3130	0.3052	0.3768	<u>0.2480</u>	0.3631	**0.2378**
	std	0.0002	0.000	0.0002	0.0018	0.0015	0.0128	0.0005

Table 7
Completion results on *Fashion-MNIST* clothing dataset: α shows missing rates, and mean and std demonstrate the average and standard deviations over 10 independent runs, respectively. The best result is highlighted in **bold**, while the <u>underline</u> indicates the second-best.

α		NMF	ANMF	AT-NMF	HQASD	DMF	ADNMF	EADNMF
0.3	mean	0.2258	0.2121	0.1815	0.2163	0.1601	<u>0.1556</u>	**0.1292**
	std	0.0007	0.000	0.0004	0.0021	0.0003	0.0005	0.0006
0.4	mean	0.2583	0.2451	0.2067	0.2285	0.1610	<u>0.1565</u>	**0.1309**
	std	0.0003	0.000	0.0003	0.0016	0.0004	0.0007	0.0003
0.5	mean	0.2918	0.2797	0.2371	0.2420	0.1614	<u>0.1569</u>	**0.1336**
	std	0.0004	0.000	0.0005	0.0012	0.0005	0.0007	0.0003
0.6	mean	0.3251	0.3143	0.2717	0.2548	0.1631	<u>0.1587</u>	**0.1382**
	std	0.0004	0.000	0.0003	0.0024	0.0006	0.0007	0.0003
0.7	mean	0.3589	0.3499	0.3109	0.2749	0.1650	<u>0.1616</u>	**0.1461**
	std	0.0003	0.000	0.0003	0.0011	0.0002	0.0005	0.0003
0.8	mean	0.3942	0.3879	0.3556	0.2968	<u>0.1694</u>	0.1716	**0.1589**
	std	0.0004	0.000	0.0002	0.0014	0.0006	0.0021	0.0003
0.9	mean	0.4264	0.4256	0.4052	0.3260	<u>0.1887</u>	0.2207	**0.1790**
	std	0.0002	0.000	0.0002	0.0013	0.0013	0.0031	0.0003

Table 8
Completion results on *MS-COCO-2017* dataset: α shows missing rates, and mean and std demonstrate the average and standard deviations over 10 independent runs, respectively. The best result is highlighted in **bold**, while the <u>underline</u> indicates the second-best.

α		NMF	ANMF	ATNMF	HQASD	DMF	ADNMF	EADNMF
0.3	mean	0.2071	0.1720	0.1582	0.1195	<u>0.1057</u>	0.1114	**0.1038**
	std	0.0002	0.0001	0.0001	0.0001	0.0001	0.0002	0.0001
0.4	mean	0.2467	0.2110	0.1810	0.1198	<u>0.1069</u>	0.1120	**0.1052**
	std	0.0002	0.0001	0.0002	0.0000	0.0001	0.0003	0.0001
0.5	mean	0.2887	0.2540	0.2127	0.1204	<u>0.1076</u>	0.1117	**0.1056**
	std	0.0003	0.0001	0.0001	0.0001	0.0002	0.0001	0.0002
0.6	mean	0.3325	0.2993	0.2533	0.1211	<u>0.1080</u>	0.1131	**0.1075**
	std	0.0002	0.0001	0.0001	0.0001	0.0002	0.0002	0.0002
0.7	mean	0.3773	0.3459	0.3035	0.1226	<u>0.1128</u>	0.1165	**0.1103**
	std	0.0001	0.0001	0.0002	0.0000	0.0003	0.0003	0.0003
0.8	mean	0.4225	0.3932	0.3626	0.1260	<u>0.1232</u>	0.1249	**0.1196**
	std	0.0001	0.0002	0.0001	0.0001	0.0004	0.0001	0.0002
0.9	mean	0.4681	0.4452	0.4326	0.1412	<u>0.1351</u>	0.1610	**0.1288**
	std	0.0001	0.0002	0.0001	0.0002	0.0003	0.0002	0.0002

removed, and then each model (standard NMF, ANMF, AT-NMF,[1] HQASD,[2] DMF,[3] or [E]ADNMF) will be trained on the remaining 10% of seen entries, and prediction performance will be then assessed. Algorithm 1 can be easily adapted to handle missing entries, as can standard NMF, by applying the binary mask $\boldsymbol{J}$ to $\boldsymbol{V}$. It should be noted that $\boldsymbol{R}_{i,j}$ is only updated for seen entries. We use the same stopping criterion for AT-NMF, and the pre-defined parameters are fair to all competing algorithms. The code to reproduce the experiments is publicly available at GitHub repository.[4]

4.4. Results

The proposed method's efficacies with/without an elastic loss will be analyzed in this part. We obtain the initial values of $\boldsymbol{W}$s and $\boldsymbol{H}$s with a shallow pre-train, then we improve them using the updating factors until the algorithm converges. Tables 2–8 present the comparison results of matrix completion algorithms for the seven datasets. The proposed algorithm is performed twice, using different loss functions, which are Frobenius and elastic loss functions.

Table 2 on *Synthetic* datasets shows that the basic model (ADNMF) and final model (EADNMF) have better results than other methods. This dataset seems simple and does not pose much of a challenge for deep models (DMF, ADNMF, and EADNMF). According to Tables 2, 3, 4, 5, and 7, it is clear that in most datasets with different degrees of observation, the proposed basic model (ADNMF) has achieved better results than comparative methods, even compared to the deep non-linear model (i.e., DMF). However, in the four more complex datasets, the face, handwriting, clothing, and object images (Tables 3, 6, 7, and 8), the ADNMF has a little poorer results than the non-linear DMF in conditions where the degree of observation is low. This means that the ADNMF, although it has better results in most cases, does not perform better than DMF in terms of generalization. But as expected, in the whole datasets, it can be seen that the final proposed method (EADNMF) has more generalizability power due to its elastic properties. Although it is a linear method with limited depth, it has better reconstruction power than a state-of-the-art non-linear method. Overall, elastic loss played a significant role in our adversarial strategy. By augmenting the adversarial DNMF with an elastic loss, the EADNMF has gained efficiency and remarkable improvement among the compared methods.

4.5. Single-image inpainting

The performance of matrix completion-based single-image inpainting is evaluated using four 300×300 RGB images [2]. The pixels are multiplied by three for each image to create a matrix of 300×900 pixels. In this study, three types of masks are applied. In the first, 50% of pixels are randomly deleted. Small and big font-size text masks are the other two masks. You can see the original and masked versions of images in Figs. 3–5. Also, these figures show that the EADNMF method has performed a visually acceptable reconstruction compared to other methods. Also, the proposed method and DMF obtain very similar results from a visual aspect. Therefore, quantitative results given in Table 9 are reported for more detailed assessments. It can be said that because there is just one sample and consequently, there is no significant learning in single-image mode, EADNMF has better results in fewer cases than in multiple-image mode. In most cases, the elastic method has better results on the random mask, which can be considered a simpler mask than used text masks, and ADNMF performs better reconstruction on *small* and *large* masks. Another noteworthy point is the reconstruction of *img*3; in this kind of image with simple patterns, the data distribution is much simpler, and it seems that the non-linear method (i.e., DMF) is overfitted to the training data. Due to the simplicity of this image (there are no considerable changes), the output results of this method are better than other methods. However, the DMF method cannot present better results on more complex images.

4.6. Convergence analysis

Fig. 6 illustrates the development of the objective function of EADNMF with train error and blind test error on the *Moffet* dataset. After every 10 minimizing iterations, an attack is applied to the model. We can see a long-term non-monotonically behavior for train loss because the Majorization-Minimization property is present in EADNMF over $\boldsymbol{R}$ and factors. As you can see, slight fluctuations are observed in the train loss that indicates the effect of the attack on the model, which in turn significantly reduces the test loss. It is worth noting that despite gaining a higher loss value on the train data, the prediction on the test data is superior. This is owing to the Intrinsic generalization of the proposed adversarial training in EADNMF. Finally, we observe that test loss is interestingly less than train loss, which makes a generalized model.

4.7. Parameter analysis

This subsection analyzes the influence of parameters on the proposed model. EADNMF has two main hyperparameters, namely λ and δ, which correspond to the attack intensity and the threshold of elastic loss, respectively. First, we experimented on the *MS-COCO-2017* dataset to examine the effectiveness of δ and the contribution of Frobenius, $\ell_{2,1}$, and the mixed

[1] https://github.com/caiting123321/AT_NMF.
[2] https://github.com/he1c/robust-matrix-completion.
[3] https://github.com/jicongfan/Matrix-completion-by-deep-matrix-factorization.
[4] https://github.com/amjadseyedi/EADNMF.

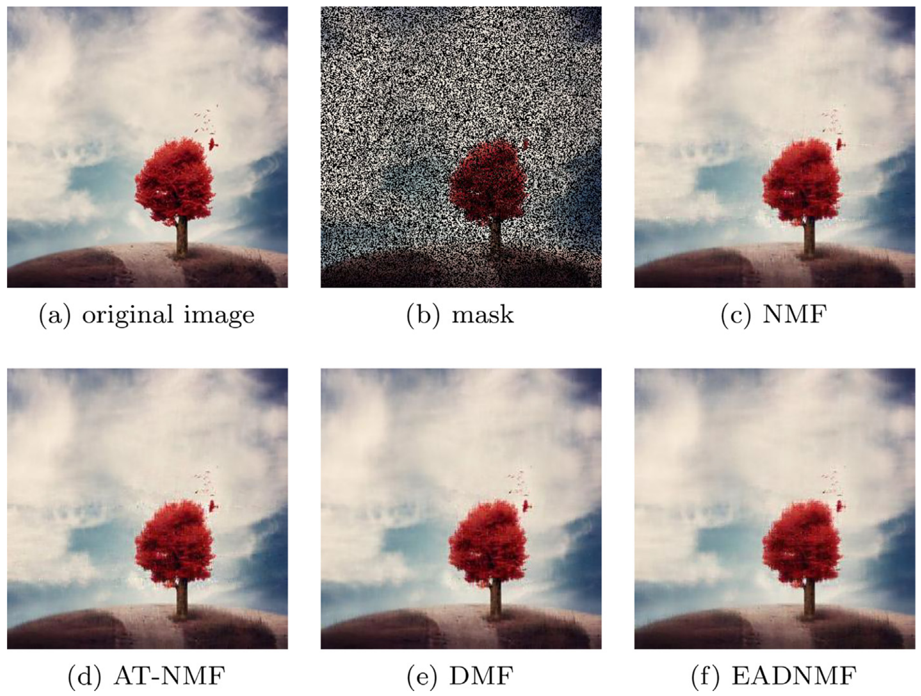


Fig. 3. Matrix Completion results for *img1* with random (50%) text mask.

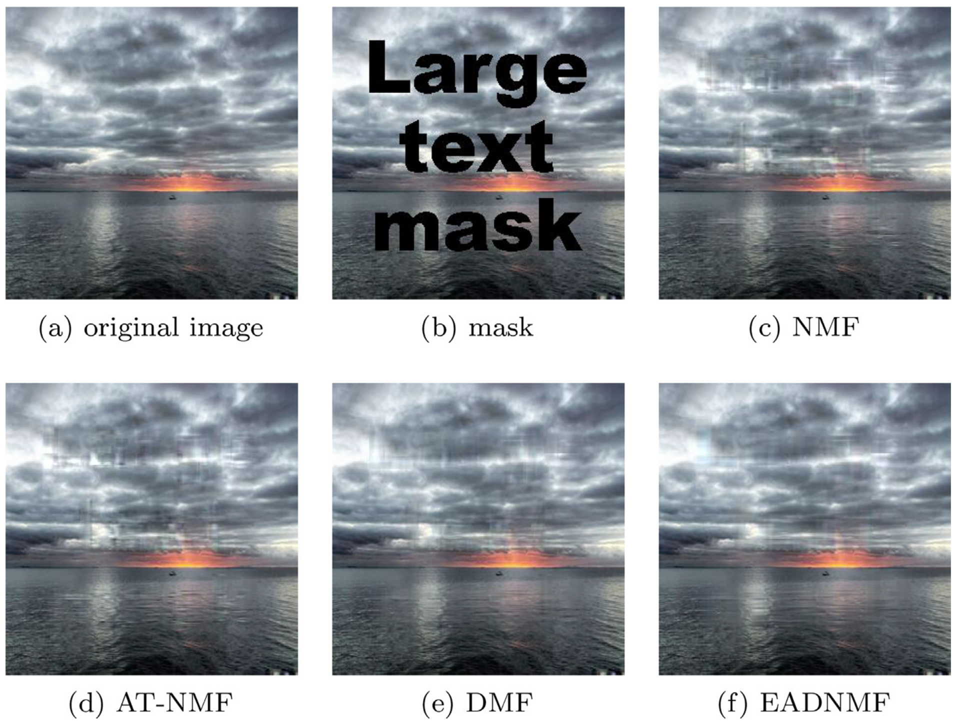


Fig. 4. Matrix Completion results for *img2* with large text mask.

loss functions. The analysis can be seen in Fig. 7. We can say that an adaptive combination of Frobenius and $\ell_{2,1}$ losses gives the lowest error. It can still be seen that, in different degrees of observation, elastic loss has an important contribution to recovery data. An $\ell_{2,1}$-based model (i.e., $\delta = 0$) is defined based on the assumption of the existence of data with outlier or

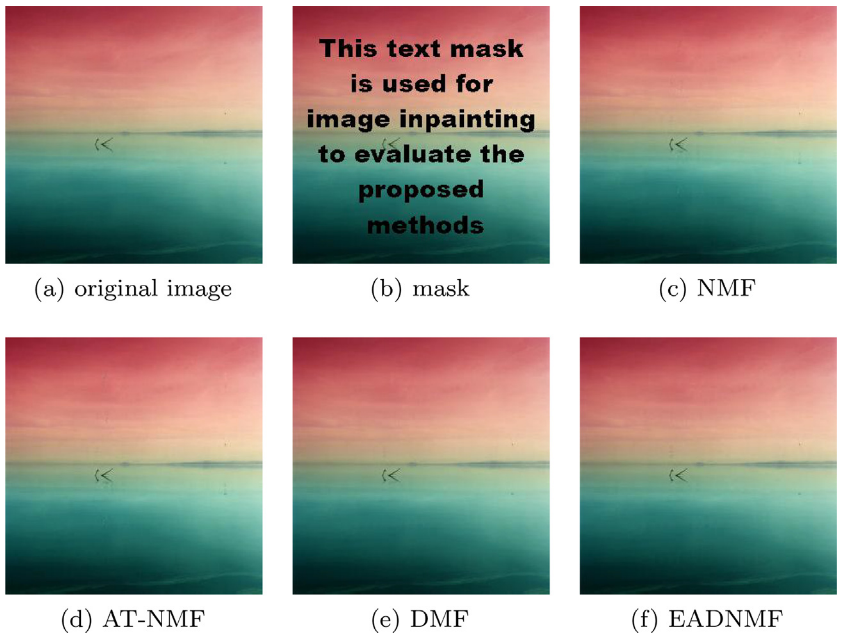

This text mask is used for image inpainting to evaluate the proposed methods
(a) original image
(b) mask
(c) NMF
(d) AT-NMF
(e) DMF
(f) EADNMF

Fig. 5. Matrix Completion results for *img3* with small text mask.

Table 9
Average RMSE on single-images: means over 10 independent runs. The best performance is highlighted in **bold**, while <u>underline</u> indicates the second-best.

image	mask	NMF	AT-NMF	DMF	ADNMF	EADNMF
img1	rand	0.0319	0.0307	**0.0268**	0.0294	<u>0.0292</u>
	small	0.0594	0.0637	0.0504	**0.0471**	<u>0.0477</u>
	large	0.1120	0.1102	0.0953	**0.0942**	<u>0.0945</u>
img2	rand	0.0248	0.0237	0.0240	0.0228	**0.0221**
	small	0.0443	0.0441	0.0374	**0.0373**	0.0374
	large	0.0804	0.0815	0.0640	**0.0595**	<u>0.0599</u>
img3	rand	0.0108	<u>0.0102</u>	0.0103	**0.0100**	**0.0100**
	small	0.0100	0.0098	**0.0084**	<u>0.0089</u>	0.0099
	large	0.0189	0.0193	**0.0169**	0.0183	<u>0.0180</u>
img4	rand	0.0307	0.0295	0.0224	0.0223	**0.0219**
	small	0.0699	0.0481	0.0528	**0.0329**	<u>0.0332</u>
	large	0.0699	0.1039	0.0518	**0.0494**	<u>0.0496</u>

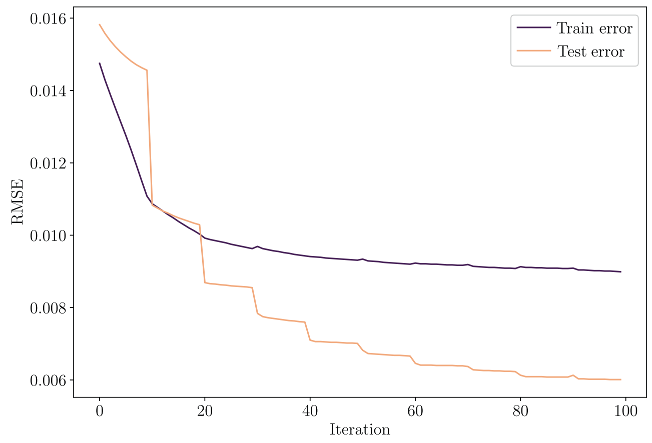


Fig. 6. Convergence of Elastic Adversarial Deep NMF (EADNMF) on the *Moffet* hyperspectral image datasets. After every 10 minimizing iterations, an attack is applied to the model.

Laplacian noise. This model behaves more cautiously in learning the whole samples than the Frobenius-based one [44]. For this reason, it can be seen that it has poor performance. On the contrary, the Frobenius-based model (i.e., $\delta = \infty$) fits all the samples under Gaussian assumption and tends to overcompensate for the influence of large deviations (outliers). It can be seen that when the number of observations is limited ($\alpha = 0.9$), the model performance is dramatically decreased. But the elastic model with an optimal combination of Frobenius and $\ell_{2,1}$ losses (i.e., $\delta = 2$) works selectively and has a flexible behavior to deal with each sample. Unlike the other two, elastic loss tries to learn clear samples precisely and learns distorted samples cautiously. We believe that the learner will eventually overfit on adversarial examples in an adversarial model with a static loss (such as Frobenius, $\ell_{2,1}$, etc.). However, an elastic loss prevents the model from overfitting and helps the model to be robust towards a mixture of multiple attacks.

Finally, to thoroughly study the two hyperparameters, we have evaluated them on all datasets through a grid-search strategy and reported one result based on the RMSE measure (*MNIST* dataset with $\alpha = 0.3$). In addition, the graph consists of about 112 evaluations referring to 7 different δs ($\delta \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$) and 16 λs ($\lambda \in \{2, 2.2, \ldots, 5\}$). It can be said that the performance of EADNMF converges to that of EDNMF as $\lambda \to \infty$, and also, EADNMF converges to that of ADNMF as $\delta \to \infty$ (sensitive to the outliers). As described in the previous section, the proposed method turns to EADNMF with $\ell_{2,1}$ when $\delta \to 0$ (insensitive to the outliers). According to Fig. 8, colors indicate the highest and the lowest values obtained on the corresponding dataset regarding the measure, shown using a spectrum of colors. The light colors represent high amounts, and the dark colors represent low amounts for the RMSE measure. Fig. 8 shows performance is achieved by elastic threshold ($\delta = 1$), and sensitivity to λ is decreased.

5. Conclusion

This paper proposed a matrix completion method called Elastic Adversarial Deep Nonnegative Matrix Factorization (EADNMF). EADNMF can recover the missing entries of complex latent structures drawn from deep latent variable models. The basic adversarial approach can produce a model that is robust towards attacks. In addition, we show that beyond adversarial robustness by an elastic loss function, EADNMF improves generalization (i.e., the test accuracy) on various data with high-degree of out-of-sample. So, training this adversarial model can provide robustness against heavy attacks. Experimental results on real-world datasets validate the effectiveness and robustness of the proposed algorithm. Nevertheless, its elastic loss is not fully-adaptive to the perturbation (or noise) degree, and choosing a suitable threshold parameter is an open challenge. In addition, most conventional factorization solvers, such as the one used in this paper, are only suitable for models with a limited number of hidden layers. Therefore, an advanced technique, such as implicit regularization, can be considered to make a deeper and possibly wider overparameterized factorization.

This work opens up new opportunities to effectively improve the generalization of modern deep models by elastic adversarial training. Furthermore, reducing the impact of noisy labels on semi-supervised models can bring additional benefits, such as improving robustness by adaptive thresholding techniques. Future work includes conducting the proposed framework on other deep models, including used models in the recommendation systems, multi-label classification, and sparse coding.

CRediT authorship contribution statement

Seyed Amjad Seyedi: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Fardin Akhlaghian Tab:** Conceptualization, Writing – review & editing, Supervision. **Abdulrahman Lotfi:** Methodology, Writing – original draft. **Navid Salahian:** Writing – original draft, Visualization. **Jovan Chavoshinejad:** Writing – original draft, Visualization.

Data availability

No data was used for the research described in the article.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Soft-elastic loss for NMF

The optimal solution of (13) is mathematically differ from equivalent variants of least square NMF and $\ell_{2,1}$-NMF. Also, we discuss the asymptotic property of the cost function,

$$\begin{cases} \beta \to 0, & z_{el}(\boldsymbol{v}^{(i)}, \boldsymbol{Wh}^{(i)}) \to z_2(\boldsymbol{v}^{(i)}, \boldsymbol{Wh}^{(i)}), \\ \beta \to 1, & z_{el}(\boldsymbol{v}^{(i)}, \boldsymbol{Wh}^{(i)}) \to z_{2,1}(\boldsymbol{v}^{(i)}, \boldsymbol{Wh}^{(i)}). \end{cases} \tag{A.1}$$

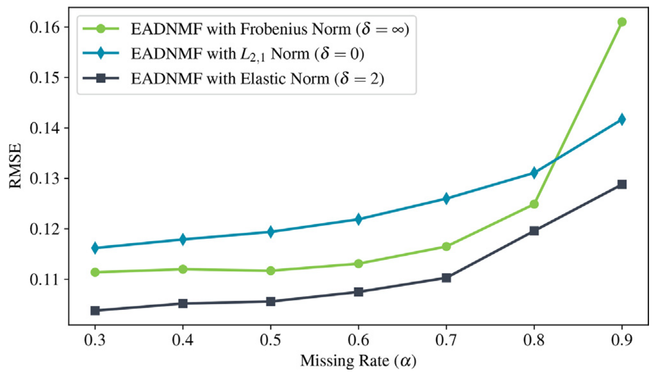


Fig. 7. Parameter analysis of Elastic Adversarial Deep NMF (EADNMF) on *MS-COCO-2017* image dataset. δ is the threshold parameter for the elastic loss and the adversary parameter is fixed to $\lambda = 2$..

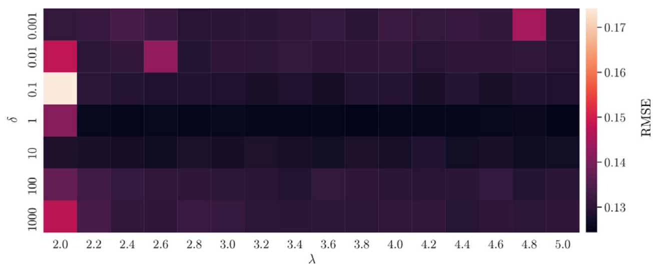


Fig. 8. Parameter analysis of Elastic Adversarial Deep NMF (EADNMF) on *MNIST* handwriting image dataset. λ is the adversary's power, and δ is the threshold parameter for the elastic loss.

When the residue $\ell_i = \|\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)}\|$ is large, $\boldsymbol{v}^{(i)}$ may have been attacked. In this case, more penalization on ℓ_1 residue loss is preferable, i.e.,

$$\ell_i \gg \delta \Rightarrow z_{el}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}) \propto z_{2,1}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}) \tag{A.2}$$

On the contrary, if the residue ℓ_i is small, $\boldsymbol{v}^{(i)}$ is clear. In this case, more penalization on ℓ_2 residue loss is preferable, i.e.,

$$\ell_i \ll \delta \Rightarrow z_{el}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}) \propto z_2(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}) \tag{A.3}$$

To realize this, β_i should closely correspond to the residue ℓ_i; however, we expect β_i can also be adjusted externally. Based on the A.2 and A.3, we define

$$\beta_i := \beta_i(\ell_i, \delta) = \frac{\ell_i - \ell_i^2}{\ell_i + \delta} \tag{A.4}$$

By multiple unique properties of (A.4), we can see the following cases.

- As $\delta \to 0, \beta_i \to 1 - \ell_i$, and therefore $z_{el}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}) \to z_{2,1}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)})$.
- As $\delta \to \infty, \beta_i \to 0$, and therefore $z_{el}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}) \to z_2(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)})$.
- As $\ell_i \gg \delta$, then $z_{el}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}) \propto z_{2,1}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)})$.
- As $\ell_i \ll \delta$, then $z_{el}(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)}) \propto z_2(\boldsymbol{v}^{(i)}, \boldsymbol{W}\boldsymbol{h}^{(i)})$.

Substitute (A.4) into (13), we obtain a closed-form of soft-elastic loss introduced as $h(.)$:

$$h_{el}(\boldsymbol{V}, \boldsymbol{W}\boldsymbol{H}) = \sum_i \frac{\delta\|\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)}\|^2}{\delta + \|\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)}\|} + \sum_i \frac{\|\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)}\|^2}{\delta + \|\boldsymbol{v}^{(i)} - \boldsymbol{W}\boldsymbol{h}^{(i)}\|} \tag{A.5}$$

References

[1] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[2] Y. Hu, D. Zhang, J. Ye, X. Li, X. He, Fast and accurate matrix completion via truncated nuclear norm regularization, IEEE Trans. Pattern Anal. Mach. Intell. 35 (9) (2013) 2117–2130.
[3] L. Yang, J. Miao, K.I. Kou, Quaternion-based color image completion via logarithmic approximation, Inf. Sci. 588 (2022) 82–105.
[4] S.A. Seyedi, S.S. Ghodsi, F. Akhlaghian, M. Jalili, P. Moradi, Self-paced multi-label learning with diversity, Asian Conference on Machine Learning 101 (2019) 790–805.
[5] X. Shu, J. Tang, G.-J. Qi, Z. Li, Y.-G. Jiang, S. Yan, Image classification with tailored fine-grained dictionaries, IEEE Trans. Circuits Syst. Video Technol. 28 (2) (2018) 454–467.
[6] K. Xu, Y. Zhang, Z. Xiong, Iterative rank-one matrix completion via singular value decomposition and nuclear norm regularization, Inf. Sci. 578 (2021) 574–591.
[7] E.J. Candès, B. Recht, Exact matrix completion via convex optimization, Found. Comput. Math. 9 (6) (2009) 717–772.
[8] C. Chen, B. He, X. Yuan, Matrix completion via an alternating direction method, IMA J. Numer. Anal. 32 (1) (2012) 227–245.
[9] J.-F. Cai, E.J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. Optim. 20 (4) (2010) 1956–1982.
[10] Z. Lin, M. Chen, Y. Ma, The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices, J. Struct. Biol. 181 (2) (2010) 116–127.
[11] X. Cao, Q. Zhao, D. Meng, Y. Chen, Z. Xu, Robust low-rank matrix factorization under general mixture noise distributions, IEEE Trans. Image Process. 25 (10) (2016) 4677–4690.
[12] Z. Wen, W. Yin, Y. Zhang, Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, Math. Programming Comput. 4 (4) (2012) 333–361.
[13] J. Tang, X. Shu, Z. Li, Y.-G. Jiang, Q. Tian, Social anchor-unit graph regularized tensor completion for large-scale image retagging, IEEE Trans. Pattern Anal. Mach. Intell. 41 (8) (2019) 2027–2034.
[14] X. Shu, J. Tang, Z. Li, H. Lai, L. Zhang, S. Yan, Personalized age progression with bi-level aging dictionary learning, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 905–917.
[15] J. Tang, X. Shu, G.-J. Qi, Z. Li, M. Wang, S. Yan, R. Jain, Tri-clustered tensor completion for social-aware image tag refinement, IEEE Trans. Pattern Anal. Mach. Intell. 39 (8) (2017) 1662–1674.
[16] L. Su, J. Liu, X. Tian, K. Huang, S. Tan, Iterative tensor eigen rank minimization for low-rank tensor completion, Inf. Sci. 616 (2022) 303–329.
[17] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[18] P. De Handschutter, N. Gillis, X. Siebert, A survey on deep matrix factorizations, Comput. Sci. Rev. 42 (2021) 100423.
[19] J. Fan, J. Cheng, Matrix completion by deep matrix factorization, Neural Networks 98 (2018) 34–41.
[20] M. Yang, S. Xu, A novel patch-based nonlinear matrix completion algorithm for image analysis through convolutional neural network, Neurocomputing 389 (2020) 56–82.
[21] J. Fan, T. Chow, Deep learning based matrix completion, Neurocomputing 266 (2017) 540–549.
[22] S. Mehrdad, M.H. Kahaei, Deep learning approach for matrix completion using manifold learning, Signal Process. 188 (2021) 108231.
[23] F. Tramèr, D. Boneh, A. Kurakin, I. Goodfellow, N. Papernot, P. McDaniel, Ensemble adversarial training: Attacks and defenses, in: 6th International Conference on Learning Representations, 2018.
[24] A. Sinha, H. Namkoong, J. Duchi, Certifying some distributional robustness with principled adversarial training, in: International Conference on Learning Representations, 2018.
[25] F. Farnia, J. Zhang, D. Tse, Generalizable adversarial training via spectral normalization, in: International Conference on Learning Representations, 2018.
[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199.
[27] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2574–2582.
[28] Z. Zhang, X. Wang, G. Lu, F. Shen, L. Zhu, Targeted attack of deep hashing via prototype-supervised adversarial networks, IEEE Trans. Multimedia 24 (2022) 3392–3404.
[29] X. Wang, Z. Zhang, B. Wu, F. Shen, G. Lu, Prototype-supervised adversarial network for targeted attack of deep hashing, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 16357–16366.
[30] L. Luo, Y. Zhang, H. Huang, Adversarial nonnegative matrix factorization, in: Proceedings of the 37th International Conference on Machine Learning, vol. 119, 2020, pp. 6479–6488.
[31] T. Cai, V.Y.F. Tan, C. Févotte, Adversarially-trained nonnegative matrix factorization, IEEE Signal Process. Lett. 28 (2021) 1415–1419.
[32] D. Meng, F. De la Torre, Robust matrix factorization with unknown noise, International Conference on Computer Vision (2013) 1337–1344.
[33] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc.: Ser. B (Statistical Methodology) 67 (2) (2005) 301–320.
[34] H. Xiong, D. Kong, Elastic nonnegative matrix factorization, Pattern Recogn. 90 (2019) 464–475.
[35] Y. Cherapanamjeri, K. Gupta, P. Jain, Nearly optimal robust matrix completion, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 797–805.
[36] O. Klopp, K. Lounici, A.B. Tsybakov, Robust matrix completion, Probab. Theory Relat. Fields 169 (1) (2017) 523–564.
[37] Y. He, F. Wang, Y. Li, J. Qin, B. Chen, Robust matrix completion via maximum correntropy criterion and half-quadratic optimization, IEEE Trans. Signal Process. 68 (2020) 181–195.
[38] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, B.W. Schuller, A deep semi-nmf model for learning hidden representations, International Conference on Machine Learning 32 (2014) 1692–1700.
[39] F. Nie, H. Huang, C. Ding, Low-rank matrix recovery via efficient schatten p-norm minimization, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 26, 2012, pp. 655–661.
[40] N. Gillis, L.T.K. Hien, V. Leplat, V.Y.F. Tan, Distributionally robust and multi-objective nonnegative matrix factorization, IEEE Trans. Pattern Anal. Mach. Intell. 44 (8) (2022) 4052–4064.
[41] S. Boyd, S.P. Boyd, L. Vandenberghe, Convex optimization, Cambridge University Press, 2004.
[42] D.R. Hunter, K. Lange, Quantile regression via an mm algorithm, J. Comput. Graphical Stat. 9 (1) (2000) 60–77.
[43] K. Ren, T. Zheng, Z. Qin, X. Liu, Adversarial attacks and defenses in deep learning, Engineering 6 (3) (2020) 346–360.
[44] D. Kong, C. Ding, H. Huang, Robust nonnegative matrix factorization using l21-norm, in: The 20th ACM International Conference on Information and Knowledge Management, 2011, pp. 673–682.
[45] Z. Zhang, Z. Lai, Y. Xu, L. Shao, J. Wu, G.-S. Xie, Discriminative elastic-net regularized linear regression, IEEE Trans. Image Process. 26 (3) (2017) 1466–1481.
[46] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[47] C. Igel, M. Hüsken, Improving the rprop learning algorithm, in: Proceedings of the second international ICSC symposium on neural computation (NC 2000), vol. 2000, 2000, pp. 115–121.
[48] D. Sheeren, M. Fauvel, S. Ladet, A. Jacquin, G. Bertoni, A. Gibon, Mapping ash tree colonization in an agricultural mountain landscape: Investigating the potential of hyperspectral imagery, 2011 IEEE International Geoscience and Remote Sensing Symposium (2011) 3672–3675.
[49] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv:1708.07747.
[50] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, European Conference on Computer Vision (2014) 740–755.