BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 66, No. 6, 2018
DOI: 10.24425/bpas.2018.125929

Predicting pairwise relations with neural similarity encoders

F. HORN[1] and K.-R. MÜLLER[1, 2, 3]*

[1]Machine Learning Group, Technische Universität Berlin, Berlin, Germany
[2]Department of Brain and Cognitive Engineering, Korea University, Seoul, Republic of Korea
[3]Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. Matrix factorization is at the heart of many machine learning algorithms, for example, dimensionality reduction (e.g. kernel PCA) or recommender systems relying on collaborative filtering. Understanding a singular value decomposition (SVD) of a matrix as a neural network optimization problem enables us to decompose large matrices efficiently while dealing naturally with missing values in the given matrix. But most importantly, it allows us to learn the connection between data points' feature vectors and the matrix containing information about their pairwise relations. In this paper we introduce a novel neural network architecture termed similarity encoder (SimEc), which is designed to simultaneously factorize a given target matrix while also learning the mapping to project the data points' feature vectors into a similarity preserving embedding space. This makes it possible to, for example, easily compute out-of-sample solutions for new data points. Additionally, we demonstrate that SimEc can preserve non-metric similarities and even predict multiple pairwise relations between data points at once.

Key words: neural networks, kernel PCA, dimensionality reduction, matrix factorization, SVD, similarity preserving embeddings.

1. Introduction

Pairwise relations, such as similarities, between data points play an important role in many areas of machine learning (ML) [1–4]. Dimensionality reduction methods such as t-SNE [5], kernel PCA (kPCA) [6], isomap [7], and locally linear embedding (LLE) [8] create low dimensional representations of data points by preserving their pairwise similarities, distances, or local neighborhoods in the embedding space, e.g., to create informative visualizations of a dataset [9]. Similarity preserving embeddings of data points can also serve as useful feature representations for other (supervised) ML tasks. For example, by computing the eigendecomposition of a kernel (i.e. similarity) matrix, kPCA projects the data into a feature space where data points can become linearly separable and noise in the data can be reduced [10–12]. In natural language processing (NLP) settings, the popular word2vec model [13, 14] learns an embedding for each word in the vocabulary by relying on the principle that similar words appear in similar contexts [15–17]. Using word embeddings as features can improve the performance in many NLP tasks such as named entity recognition or text classification [18–20]. The prediction of pairwise relations themselves is at the heart of important real world ML applications such as the prediction of whether or not a drug could interact with a certain protein [21] or for recommender systems, where the task is to predict the rating a user would give to a certain item [22, 23] or to identify similar items that could be promoted alongside an item of interest [24]. Another active research area is concerned with the analysis of graphs, such as social networks, where the pairwise relations between nodes are of key importance [25, 26].

Such pairwise relations between data points can be represented as a rectangular matrix $R \in \mathbb{R}^{m \times n}$, which could, for example, contain the ratings of m items by n users. In the following, we will primarily focus on pairwise similarities between m data points, stored in a square symmetric matrix $S \in \mathbb{R}^{m \times m}$, but also discuss how our results generalize to arbitrary pairwise relations R.

In this paper, we introduce our novel neural network architecture called similarity encoder (SimEc), which learns (low dimensional) similarity preserving embeddings for data points. To be more precise, SimEc represent the data in such a way that the scalar product between the embedding vectors $\mathbf{y}_i, \mathbf{y}_j \in \mathbb{R}^d$ of two data points approximates their similarity, i.e., $\mathbf{y}_i \mathbf{y}_j^\top \approx S_{ij}$. Furthermore, given some original (high dimensional) feature vectors $\mathbf{x}_i \in \mathbb{R}^D \forall i \in \{1, \ldots, m\}$, a SimEc additionally provides a linear or non-linear mapping function $f' : \mathbf{x}_i \to \mathbf{y}_i$, which can be used to project new data points into the similarity preserving embedding space, i.e., to compute out-of-sample (OOS) solutions. Here it is important to note that these feature vectors do not have to be directly related to the given target similarities stored in S, in fact, we do not need to know how S was computed at all, e.g., it could also contain human similarity ratings. Furthermore, SimEc can deal with missing values in the similarity matrix S, can embed data points based on metric or non-metric similarities, and can be used to predict multiple pairwise similarities or other relations between data points at once. We provide a keras [27] based Python implementation of the model, which can be trained efficiently on GPUs[1].

*e-mail: klaus-robert.mueller@tu-berlin.de

Manuscript submitted 2018-04-18, initially accepted for publication 2018-05-16, published in December 2018.

[1] https://github.com/cod3licious/simec

After relating our model to previous work, we detail its architecture in Section 2 and then demonstrate its effectiveness in multiple experiments (Section 3) before concluding the paper with a discussion.

1.1. Related work. The optimal (in a least squares sense) low dimensional embeddings to factorize a matrix R or S can be found by computing a singular value decomposition (SVD) or eigendecompositon of the matrix and using the d largest eigenvalues and corresponding eigenvectors to compute a low rank approximation of the matrix. However, performing an SVD is computationally very expensive for large matrices, and in these cases requires the use of approximate iterative methods [22]. Furthermore, an exact decomposition can not be computed for matrices that contain missing values, in which case weighted error functions need to be employed [28]. Back in 1982, a simple neural network (NN) was conceived to compute a PCA [29] and in 1992, NNs were proposed as a method to efficiently compute the SVD [30] or eigendecomposition [31] of a matrix while naturally dealing with missing values in the target matrix, which we will discuss in more detail in Section 2.1.

If a similarity matrix was computed with a known support vector kernel function [4, 11, 32], a manually devised, kernel-specific random mapping from the original input to the kernel feature space could be used to create similarity preserving embeddings for large datasets very efficiently [33]. By interpreting this mapping as a neural network, it can be further fine-tuned to the dataset at hand [34]. But while spectral methods such as kPCA provide optimal similarity preserving embeddings based on the pairwise similarities for a given set of data points, a critical issue remains, namely that they can only compute OOS solutions, i.e., embeddings for new tests points, if their similarity to the original training examples can be computed with a known kernel function [35]. SimEc on the other hand learn a direct mapping from the original feature space to the similarity preserving embedding space and therefore do not require knowledge of how the pairwise similarities were computed (Fig. 1). It would be possible to train an additional regression model to learn the mapping from the original input feature space to the embeddings computed by the spectral method [36, 37]. However, the best similarity preserving embeddings that can be realized as a transformation of the original feature vectors might not necessarily correspond to the embeddings found by the spectral method, thereby losing some information by the mapping, resulting in unnecessarily poor similarity approximations [38][2].

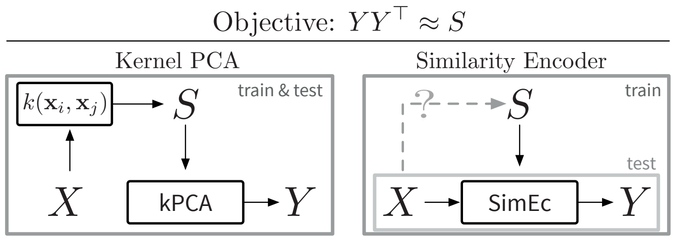

Fig. 1. Kernel PCA and SimEc both aim to project the data points into an embedding space where the target similarities can be approximated by the scalar product of the embedding vectors, but kernel PCA also needs to compute a kernel map, i.e., the similarities to the training data points, to be able to embed new test samples

Previous work concerned with simultaneously factorizing a matrix with pairwise relations while learning a mapping from the original input space to the low dimensional embedding space can be categorized into the two approaches outlined below. In both cases, the mapping from the input to the embedding space is usually realized by neural networks, as they provide the flexibility to learn arbitrary functions.

1.1.1. Embedding with a single NN. In the first approach, a single neural network is trained to map the points into a similarity preserving embedding space by computing the embeddings for a batch of training samples and then comparing the pairwise similarities (or distances) of these embedded points against the target similarities to compute the error used in the backpropagation procedure to tune the network's parameters. With this approach, extensions for t-SNE [39] and other classic manifold learning methods [40, 41] were developed, which enable the computation of OOS solutions. A particularly interesting realization of this approach are deep kernelized autoencoders [42], which train an autoencoder network with an additional objective to not only minimize the reconstruction error of the data points themselves but also the mismatch between the dot product of a batch of embedding vectors and the corresponding block from a kernel matrix. The decoder part of the autoencoder network thereby also provides a mapping from the embedding space back to the original feature space, which can be used to compute the pre-image of an embedding vector [10]. By directly minimizing $\|S - YY^\top\|$, these methods successfully learn similarity preserving embeddings, however, because they always operate on batches of points, these methods scale quadratically and efficient training is highly dependent on the choice of the batch size, requiring either lots of memory or many combinations of randomly chosen samples to cover all pairwise similarities. In an effort to improve on this, the method of auxiliary coordinates can be used to train a NN in an alternating fashion, in one step optimizing the mapping from the input to the embedding space, in the other step improving the similarity approximation of the embedding itself [38]. As we will see in Section 2.2, while the weight matrix of the last layer of the SimEc architecture could be interpreted as a set of auxiliary coordinates as well, training a SimEc network does not require alternating steps in the optimization procedure.

The above mentioned methods all learn embeddings based on pairwise similarities, but can *not* generalize to other types

[2] For example, eigenvalue d might only be slightly larger than eigenvalue $d+1$, however, the dth eigenvector might contain information that is not present in the original feature vectors, while the information encoded in eigenvector $d+1$ can be preserved by a transformation of the feature vectors. By learning the mapping and factorization together, it is possible to create a d-dimensional embedding that instead retains the information from the $d+1$ eigenvalue and vector, thereby resulting in an only slightly worse approximation of S compared to the spectral method, while not losing any accuracy in the mapping step.

of pairwise relations, specifically those involving two different kinds of data, e.g., ratings of items by users.

1.1.2. Learning two mappings into the same embedding space. This brings us to the second approach, where two networks are trained to simultaneously map two (different kinds of) input feature vectors into the same embedding space. Again, the objective is for the similarity between the two embedding vectors to approximate the respective pairwise relation stored in the target matrix R. For pairwise similarities, the mapping into the embedding space can also be realized by a siamese network, i.e., two networks with shared parameters operating on the same kinds of input data [43]. As these networks operate on pairs of samples, training again scales quadratically, but here at least the batch sizes for both networks can be chosen independently. Furthermore, this approach is often employed for recommender systems [44], where the target matrix R is typically very very sparse, which means by training only on pairs of samples with known targets, the training time can be greatly reduced. In these cases, often some form of negative sampling is employed during training to consider for every positive sample pair (e.g. a song a user has listened to) some negative pairs (songs a user has not listened to) as well, as these can provide additional information [45]. With a fast training procedure for target matrices where the number of non-zero elements is much smaller than $m \cdot n$ and the benefit of learning multiple mapping functions simultaneously for projecting different kinds of feature vectors into the same embedding space, this second approach is very useful for many applications scenarios. SimEc, on the other hand, are designed to efficiently factorize dense matrices (while being able to handle missing values in the target matrix) and, while they rely on only a single neural network to map input features into a similarity preserving embedding space, we discuss in the next section how they can be trained to *predict multiple pairwise relations at once* based on the same embedding.

2. The SimEc model

In the following, we will first describe how neural networks can realize the computation of an SVD of a rectangular matrix $R \in \mathbb{R}^{m\times n}$ [30] and the eigendecomposition of a square symmetric matrix $S \in \mathbb{R}^{m\times m}$ [31]. Then we detail how these models can be extended to arrive at the SimEc neural network architecture.

2.1. Matrix factorization with neural networks. With singular value decomposition (SVD), a matrix $R \in \mathbb{R}^{m\times n}$ can be decomposed as

$$R = U\Sigma V^\top,$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ contain the eigenvectors of $RR^\top$ and $R^\top R$ respectively while the corresponding eigenvalues are stored in$\Sigma \in \mathbb{R}^{m\times n}$. By using only the d largest eigenvalues and corresponding eigenvectors, a low rank approximation of R can be obtained, i.e., $R \approx U_{[:,:d]}\Sigma_{[:d,:d]}V^\top_{[:d,:]}$.

By setting $W_1 = U_{[:,:d]}\sqrt{\Sigma_{[:d,:d]}}$ and $W_2 = \sqrt{\Sigma_{[:d,:d]}}V^\top_{[:d,:]}$, the low rank approximation of R can be rewritten as

$$R \approx W_1W_2 = I_mW_1W_2,$$

where $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix.

A simple feed forward neural network $f(\mathbf{x}_i)$ can now be constructed with two layers defined by the weight matrices $W_1 \in \mathbb{R}^{m\times d}$ and $W_2 \in \mathbb{R}^{d\times n}$ and without any non-linear activation functions. Given some input vector $\mathbf{x}_i \in \mathbb{R}^m$, the first layer computes

$$f'(\mathbf{x}_i) = \mathbf{x}_iW_1 = \mathbf{y}_i,$$

where we call $\mathbf{y}_i \in \mathbb{R}^d$ the embedding of the ith data point $\mathbf{x}_i$, with $i \in \{1, \ldots, m\}$. With both layers, the network computes

$$f(\mathbf{x}_i) = f'(\mathbf{x}_i)W_2 = (\mathbf{x}_iW_1)W_2 = \mathbf{y}_iW_2 = \hat{\mathbf{r}}_i, \tag{1}$$

the n-dimensional vector $\hat{\mathbf{r}}_i$. Expressed in matrix notation, given an input matrix $X \in \mathbb{R}^{m\times m}$ the network first computes an embedding matrix $Y \in \mathbb{R}^{m\times d}$ and from it the output $\hat{R} \in \mathbb{R}^{m\times n}$:

$$f(X) = f'(X)W_2 = (XW_1)W_2 = YW_2 = \hat{R}.$$

If the network is now trained (with backpropagation) to minimize the mean squared error of its output to a target matrix R, i.e.

$$\min \|R - f(X)\|_F^2,$$

while we use as input to the network the identity matrix, i.e., $X = I_m$, then, once the weights of the network have converged to a local optimum, W_1W_2 is a low rank approximation of the matrix $R \in \mathbb{R}^{m\times n}$ [30].

When computing an SVD of a matrix, the eigenvectors stored in the matrices U and V are orthogonal (i.e. $V^\top V = I_n$), which can be added as a further constraint to the cost function:

$$\min \|R - I_mW_1W_2\|_F^2 + \lambda\|I_dW_2W_2^\top - W_2W_2^\top\|_F^2,$$

where λ is a hyperparameter to control the strength of this regularization[3].

Should R contain missing values, then the error used in the backpropagation procedure to tune the network's parameters is only computed considering the available entries of the matrix. In this case especially it is advisable to additionally use other regularization techniques such as adding ℓ_2 regularization terms to the cost function.

As the decomposition of a square symmetric matrix $S \in \mathbb{R}^{m\times m}$ into its eigenvalues and vectors is a special case of an SVD

[3] While this will encourage orthogonal rows in W_2, since the rows do not need to have unit length, the values on the diagonal of $W_2W_2^\top$ should not be penalized. This kind of regularization is usually only necessary if d is chosen to be greater than the number of significant eigenvalues. Please note that the rows of W_2 are not necessarily ordered by the magnitude of the corresponding eigenvalues.

(where $V = U$), the same NN can be used, only with an additional constraint to learn a symmetric factorization, i.e., to encourage $I_m W_1 = Y = W_2^\top$. This can be achieved with the cost function

$$\min \left\| S - I_m W_1 W_2 \right\|_F^2 + \lambda \left\| S - W_2^\top W_2 \right\|_F^2,$$

where, after convergence, $YW_2 \approx W_2^\top W_2 \approx YY^\top \approx S$. This also results in the same eigenvector based embedding $Y \in \mathbb{R}^{m \times d}$ as found by kernel PCA.

2.2. Similarity encoders. Now that the factorization of a matrix R or S is expressed in terms of optimizing a neural network, this setup can be further extended to yield our SimEc architecture. In particular, the first linear layer of the neural network, $f'(\mathbf{x}_i) = \mathbf{x}_i W_1 = \mathbf{y}_i$, can be replaced by any kind of (deep) neural network to map arbitrary feature vectors $\mathbf{x}_i \in \mathbb{R}^D \forall i \in \{1, \ldots, m\}$ into the low dimensional embedding space (Fig. 2). Equation 1 then becomes

$$f(\mathbf{x}_i) = f'(\mathbf{x}_i) W_l = \mathbf{y}_i W_l = \hat{\mathbf{r}}_i,$$

where again $\mathbf{y}_i \in \mathbb{R}^d$ is the embedding of the ith data point $\mathbf{x}_i$ and $W_l \in \mathbb{R}^{d \times n}$ is the weight matrix of the last (linear) layer of the full network $f(\mathbf{x}_i)$, while $f'(\mathbf{x}_i)$ could, for example, be a convolutional neural network (CNN) mapping images into the embedding space. This similarity encoder network is again trained to minimize $\left\| R - f(X) \right\|_F^2$, thereby learning the factorization $R \approx f'(X) W_l = YW_l$.

If the output of the SimEc should always be in a specific range, e.g., if the target matrix R contains star ratings from 1 to 5, it may be beneficial to add an additional non-linearity after computing $f'(\mathbf{x}_i) W_l$ to ensure the predicted values are within this range. However, there should not be any non-linearity at the last layer of $f'(\mathbf{x}_i)$ as the embedding values $\mathbf{y}_i$ should be able to assume unconstrained values.

Regularization terms can again be added to the cost function as discussed before. However, it should be noted that the constraint to encourage a symmetric factorization of a similarity matrix S, i.e., the regularization term $\left\| S - W_l^\top W_l \right\|_F^2$, can significantly increase the computational complexity of the optimization procedure, as computing $W_l^\top W_l$ scales with m^2. However, in practice it is often enough to only train with a subsample of S using $n \ll m$ targets, i.e., optimizing

$$\min \left\| S_{[:, :n]} - f'(X) W_l \right\|_F^2 + \lambda \left\| S_{[:n, :n]} - W_l^\top W_l \right\|_F^2,$$

with $W_l \in \mathbb{R}^{d \times n}$ and $f(X) = \hat{S} \in \mathbb{R}^{m \times n}$, which greatly reduces the overall complexity and memory requirements of the training procedure. Even though the number of targets in the output is reduced, all m training examples can still be used as input during training.

Instead of limiting the number of targets, it might also be worth considering whether it is necessary to enforce a sym-

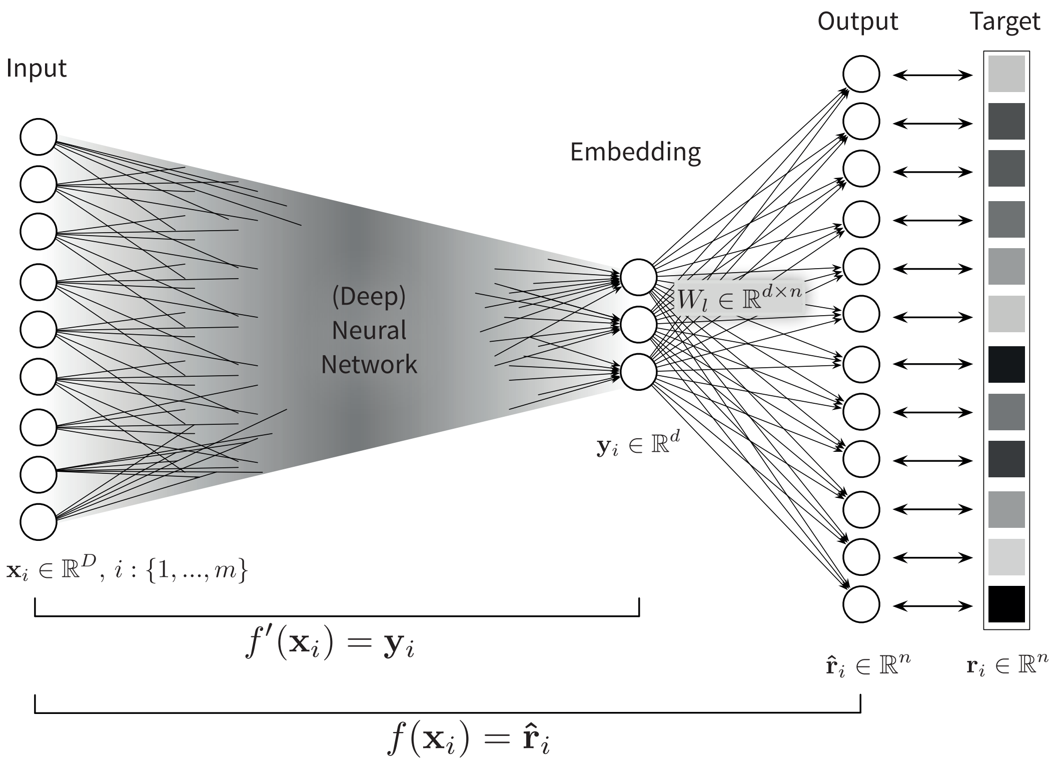
Input
Output
Target
Embedding
(Deep)
Neural
Network
$W_l \in \mathbb{R}^{d \times n}$
$\mathbf{y}_i \in \mathbb{R}^d$
$\mathbf{x}_i \in \mathbb{R}^D,\ i : \{1, \ldots, m\}$
$f'(\mathbf{x}_i) = \mathbf{y}_i$
$\hat{\mathbf{r}}_i \in \mathbb{R}^n$
$\mathbf{r}_i \in \mathbb{R}^n$
$f(\mathbf{x}_i) = \hat{\mathbf{r}}_i$

Fig. 2. Similarity encoder (SimEc) architecture. A (deep) neural network, $f'(\mathbf{x}_i)$, is used to map the original feature vector $\mathbf{x}_i \in \mathbb{R}^D$ to an embedding $\mathbf{y}_i \in \mathbb{R}^d$. This embedding is then multiplied by another weight matrix $W_l \in \mathbb{R}^{d \times n}$, which corresponds to the last layer of the full SimEc network $f(\mathbf{x}_i)$, to compute $\hat{\mathbf{r}}_i \in \mathbb{R}^n$, i.e., the approximation of one row of the target matrix $R \in \mathbb{R}^{m \times n}$. After the SimEc is trained to minimize $\left\| R - f(X) \right\|_F^2$ given the full feature matrix $X \in \mathbb{R}^{m \times D}$, it then computes a rank d approximation of R as $f(X) = f'(X) W_l = YW_l$

metric factorization of S $\left(\text{as } YW_l \approx W_l^\top W_l \approx YY^\top\right)$ at all. If the SimEc only needs to predict the similarities between a new sample and the existing samples or even just between the existing samples themselves, e.g., to fill missing values in S, then the regularization term can in practice be ignored. The similarities between a new sample $\mathbf{x}_j$ and the training samples can then be computed as

$$f(\mathbf{x}_j) = f'(\mathbf{x}_j)W_l = \mathbf{y}_j W_l = \hat{\mathbf{s}}_j \in \mathbb{R}^m,$$

instead of $f'(\mathbf{x}_j)f'(X)^\top = \mathbf{y}_j Y^\top$.

A similar choice should be made when factorizing a rectangular matrix R. By default a SimEc only learns the mapping from one input feature space to the embedding space and then predicts the values of R by multiplying this embedding with W_l. This is sufficient in many cases. For example, for an established social network site, thousands of pieces of new content are uploaded every second and at the same time older content becomes irrelevant, while the user base remains fairly constant. In such a scenario it might be sufficient to simply predict which users might be interested in a new piece of content, which can be done by using the full SimEc network to predict $f(\mathbf{x}_i) = \hat{\mathbf{r}}_i$ for some content feature vector $\mathbf{x}_i \in \mathbb{R}^D$. Nevertheless, it is also possible to train a second SimEc network to additionally project the set of n users into the same embedding space as some m items, thereby making it possible to predict ratings for both new items *and* new users as the scalar product of their embedding vectors. For this, a SimEc network f_1 is first trained on one set of feature vectors $X_1 \in \mathbb{R}^{m\times D}$ to approximate R (or a subset of it). After the training is complete, these feature vectors are projected into the embedding space to yield $f_1'(X_1) = Y_1 \in \mathbb{R}^{m\times d}$. Then, a second SimEc f_2 can be trained using the second set of feature vectors $X_2 \in \mathbb{R}^{n\times P}$ to approximate $R^\top$ (or again a subset of it), only that in this case the weights of the last layer are kept fixed as $W_l = Y_1^\top$. Both SimEcs together then provide mapping functions for two different kinds of input feature vectors into the same embedding space such that $f_1'(X_1)f_2'(X_2)^\top = Y_1 Y_2^\top \approx R$.

2.2.1. Preserving non-metric similarities and predicting multiple pairwise relations at once. Non-metric similarities are characterized by an eigenvalue spectrum with significant negative eigenvalues. Spectral embedding methods such as kPCA require positive semi-definite similarity matrices to compute the low dimensional embedding of the data and would in this case discard the information associated with the negative eigenvalues. However, Laub et al. [46] have shown that this negative part of the eigenvalue spectrum can reveal interesting features in the data and therefore should not be ignored.

A non-metric similarity matrix S is equal to the difference between two similarity matrices S_1 and S_2, where S_1 has the same p positive eigenvalues as S, while the non-zero eigenvalues of S_2 correspond to the q negative eigenvalues of S. Correspondingly, a factorization of S into $YY^\top$ would need to capture the relation between S_1 and S_2, i.e.,

$$S = S_1 - S_2 \approx YY^\top = Y_p Y_p^\top - Y_q Y_q^\top.$$

However, the only way to get this negative part of the product $YY^\top$ would be for the values of Y_q to be imaginary, which is generally not desirable for such embeddings.

With SimEcs it is nevertheless possible to approximate a non-metric similarity matrix S. Since during training S is approximated as $f'(X)W_l = YW_l$ and not $YY^\top$, some parts of Y and W_l can have opposite signs, which makes it possible to not only approximate S_1 but also $(-S_2)$. In this case the regularization term $\left\|S - W_2^\top W_2\right\|_F^2$ would be counterproductive[4].

SimEc can also be trained explicitly to preserve the information provided by multiple similarity matrices $S_1, \ldots, S_k$. The easiest way to do this is to simply compute the average of these similarity matrices and then train a SimEc as before on this averaged S. However, because SimEcs preserve the information associated with the d largest eigenvalues, the embedding only captures all k similarities if the largest eigenvalues of the k similarity matrices are equal. Therefore, before computing their average, the similarity matrices should first be normalized by dividing them each by their respective largest eigenvalue.

If the focus is not on the similarity preserving embedding itself, but rather it is important to accurately predict multiple similarities or other pairwise relations at the same time, then the SimEc network can be extended to have multiple last layers, i.e., by choosing $W_l \in \mathbb{R}^{d\times n\times k}$ a SimEc can compute

$$f(X) = f'(X)W_l = YW_l = \hat{R} \in \mathbb{R}^{m\times n\times k}.$$

Similarly, in addition to a last layer W_l, the SimEc network can also be extended by a mirrored version of $f'(\mathbf{x}_i)$, thereby adding a decoder part to the network, which can be used to compute the pre-image of an embedding like in the deep kernelized autoencoder networks [42].

3. Experiments and results

In the following, we demonstrate that SimEc can learn a mapping from an original input feature space into a similarity preserving embedding space, even if the target similarities were not computed from the original feature vectors. Furthermore, we discuss the influence of regularization and the number of targets on the embedding quality, as well as show that SimEc can create a faithful embedding even if the target similarity matrix contains over 90% missing values. Finally, we demonstrate that SimEc can predict non-metric similarities and multiple similarities at once.

As SimEcs simultaneously factorize a similarity matrix and learn a mapping into the similarity preserving embedding space, the most appropriate method to compare a SimEc's performance with is the combination of the eigendecomposition of S, to get optimal similarity preserving embeddings, and an additional regression model, trained to learn the mapping from the orig-

[4] It should be noted that a d-dimensional SimEc embedding generally captures the information associated with the d eigenvalues with the largest absolute values; should the magnitude of the largest negative eigenvalue be smaller than the first d positive values, then this information will still be ignored.

inal feature space to the embedding space. As the embeddings produced by the regression model will at most be as good as the original embeddings created by decomposing S [38], in most experiments we only report the optimal performance achieved by the eigendecomposition as a reference.

Further details as well as the code to replicate these experiments and more is available online [47].

3.1. Dataset. All experiments are performed on subsets of the MNIST dataset, which contains 28×28 pixel images depicting handwritten digits. For the first set of experiments, we randomly subsampled 10 k images from all classes, of which 80% are assigned to the training set and the remaining 20% to the test set. For the second set of experiments, we randomly subsampled 5 k images depicting zeros and sevens and we refer to this as the "MNIST 0/7" dataset.

As input feature vectors we use the 784 pixel values of each image, which we normalize by their maximum value and center to have zero mean. The respective target similarity matrices were also centered (as it is being done for kPCA as well [11]) and, if necessary, normalized to be in the range $[-1, 1]$.

3.2. Mapping into a similarity preserving embedding space. To demonstrate that SimEc can learn the connection between data points' feature vectors and an unrelated target similarity matrix S, we compute pairwise similarities between the MNIST images based on their class labels. This similarity matrix is 1 for a pair of images depicting the same digit and 0 elsewhere. With increasing embedding dimensionality d, the mean squared error between the target similarity matrix S and its approximation $\hat{S}$, computed as the dot product of the embedding vectors, $YY^\top$, should decrease. The eigendecomposition of S provides the optimal similarity preserving embeddings. However, this does not provide a mapping from the original input feature space to the embedding space to compute OOS solutions, as for new test samples the class based similarities are not available. As shown in Fig. 3, the embeddings produced by a linear SimEc, where $f'(\mathbf{x}_i)$ consists of only a single linear layer mapping the input

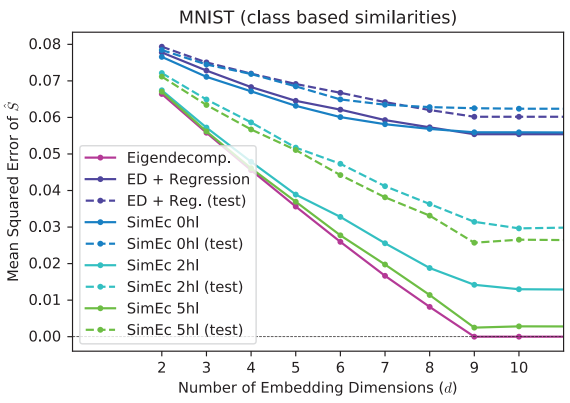

Fig. 3. Mean squared errors between the target similarity matrix S and its approximation $\hat{S}$, computed as the dot product of the embedding vectors, $YY^\top$, with increasing embedding dimensionality d

vectors into the embedding space, are comparable to those of a linear ridge regression model that learned the connection between the feature vectors and the embeddings produced by the eigendecomposition of S. By using a SimEc with a deeper NN $f'(\mathbf{x}_i)$ with several non-linear hidden layers to map the feature vectors into the embedding space, the error of the approximation gets very close to that of the eigendecomposition.

3.3. Of hyperparameters and missing values. Next, we investigate the influence of hyperparameter choices and missing values in the target similarity matrix. For this, a SimEc with one additional hidden layer is trained to create ten dimensional embeddings to approximate an RBF kernel matrix. Corresponding embeddings created with kernel PCA serve as a reference.

First, we analyze the influence of the regularization term $\lambda\left\|S - W_l^\top W_l\right\|_F^2$ (Fig. 4 left panel). While the output of the SimEc network, YW_l, always faithfully approximates the target similarities, the dot product of the embedding vectors, $YY^\top$, only achieves similar accuracies when a symmetric factorization of S is enforced.

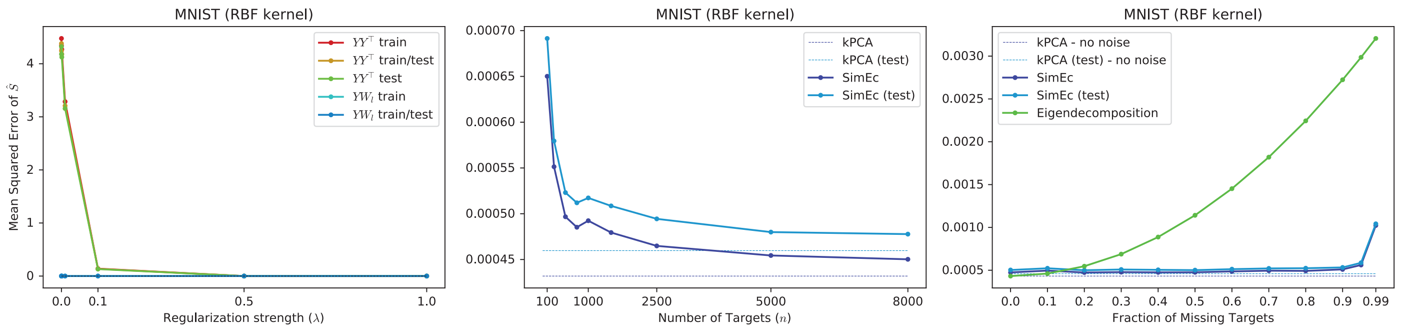

Fig. 4. Left: Importance of the regularization term $\lambda\left\|S - W_l^\top W_l\right\|_F^2$ to ensure not only the output of the SimEc network, YW_l, approximates the target similarity matrix, but also the dot product of the embedding vectors, $YY^\top$. With YW_l it is only possible to predict the similarities between new samples and those used for training the network, while with $YY^\top$ the similarities between new test samples can be computed as well. Middle: Even if only a fraction of targets is used for training, the mean squared error between $YY^\top$. and S is close to the optimal error achieved by kernel PCA. Right: Influence of missing values in the target similarity matrix. Kernel PCA computed on the full matrix again serves as the optimal reference error, while the green curve depicts the error achieved by computing the eigendecomposition of the matrix where the missing values were filled with the mean of the matrix

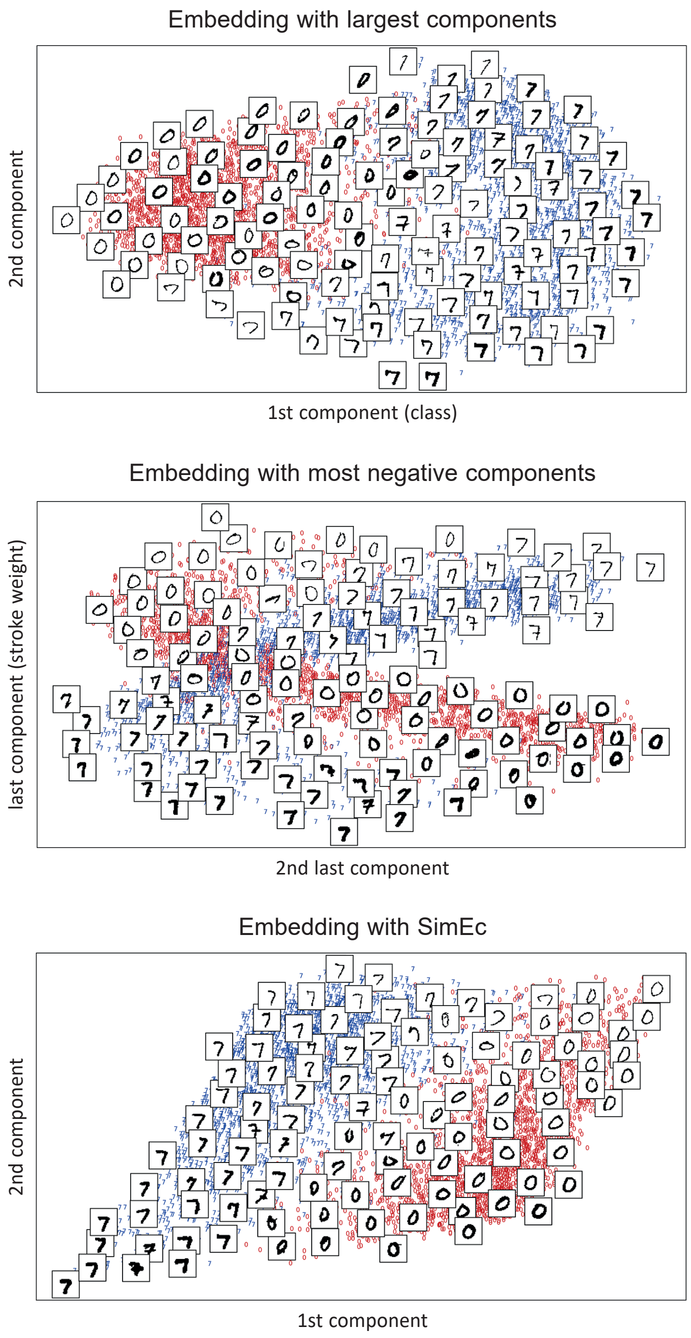

Fig. 5. Embedding of the MNIST 0/7 dataset based on the largest (top) and most negative (middle) eigenvalues of the Simpson similarity matrix, as well as a SimEc embedding of dimensionality $d = 2$ based on the sum of the similarity matrices associated with the largest and most negative eigenvalues (bottom)

As we discussed before, this regularization dramatically increases the computational complexity and memory requirements of the training procedure, as it scales quadratically with the output dimensionality. However, often only a fraction of the targets is required for $YY^\top$ to approximate S reasonably well (Fig. 4 middle).

As pairwise data can be expensive to collect or be systematically unavailable (e.g. in movie ratings), target matrices will often contain many missing values. An exact eigendecomposition of a matrix with missing values can not be computed, and instead these entries in the matrix need to be filled, e.g., by the mean of the given targets. However, this results in an almost linear increase in the mean squared error between the full target matrix and the approximation computed as $YY^\top$ (Fig. 4 right panel). With the embeddings created with SimEc, on the other hand, the target similarities can be faithfully approximated even if the target matrix contains over 90% missing values.

3.4. Predicting non-metric similarities and more. In the following experiments we demonstrate that SimEc can predict non-metric similarities and multiple similarities at once. For this we use the MNIST 0/7 dataset and compute the target similarity matrix S using the Simpson similarity score on binarized feature vectors:

$$S_{ij} = \frac{\#\{\text{pixels that are black in both } i \text{ and } j\}}{\min\{\#\{\text{black pixels in } i\}, \#\{\text{black pixels in } j\}\}}.$$

As previously shown by Laub et al. [46], the eigenvalue spectrum of this matrix contains significant negative eigenvalues and embeddings based on the corresponding eigenvectors reveal interesting features. While the embedding based on the largest eigenvalues separates the data points by class (Fig. 5 top), an embedding based on the most negative eigenvalues sorts the images by stroke weight (Fig. 5 middle). SimEc are able to create embeddings based on non-metric similarities as well. While the embedding learned by a SimEc (with one hidden layer) captures the features associated with the negative eigenvalues, their dot product would not optimally approximate S, as for this the dimensions associated with the negative eigenvalues would have to be imaginary. However, by computing $\hat{S} = YW_l$ the non-metric similarities can be predicted quite well (Fig. 6), with errors closer to those of the embeddings based

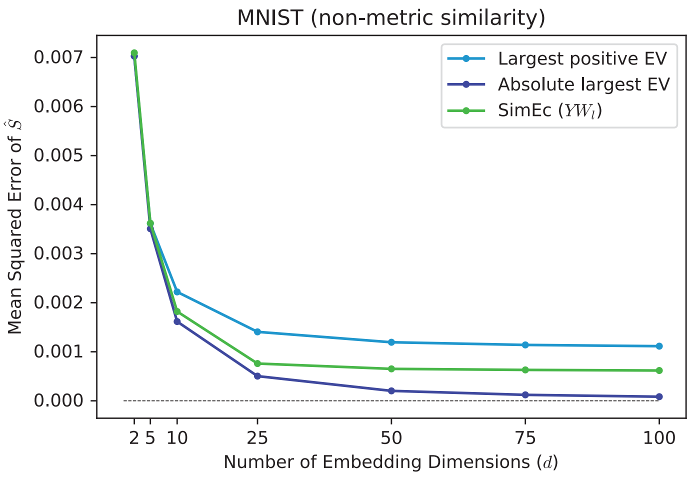

Fig. 6. Mean squared errors of the non-metric similarity matrix S and the dot product of the embeddings based on the largest positive eigenvalues, the embeddings based on the largest absolute eigenvalues (where dimensions associated with negative eigenvalues were cast as imaginary numbers), and the prediction of S with a SimEc as YW_l

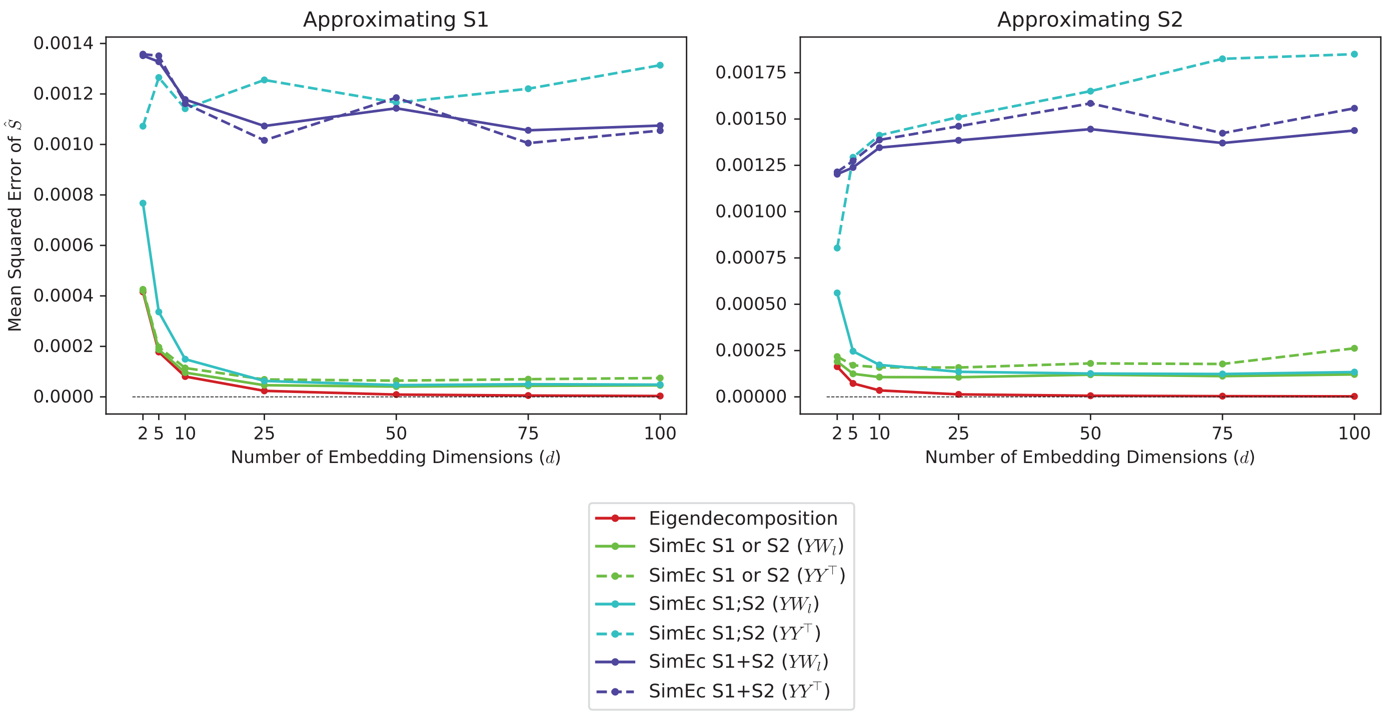

Fig. 7. Mean squared errors when approximating either S_1 (left) or S_2 (right). The eigendecomposition of the respective matrix yields the optimal similarity preserving embedding. Depicted in green are the errors achieved with a SimEc trained to approximate either S_1 or S_2 alone; shown in cyan are the errors achieved with a SimEc trained to approximate the tensor containing the stacked matrices S_1 and S_2; while the purple curves show the errors achieved with a SimEc trained to approximate the matrix $S_1 + S_2$. Continuous lines depict the prediction of S as YW_l, while dashed lines correspond to the approximation as $YY^\top$

on both positive and negative eigenvalues instead of those of the embeddings based only on the largest positive eigenvalues (i.e. a regular kPCA embedding).

As discussed in the previous section, the non-metric similarity matrix can be decomposed as $S = S_1 - S_2$, where S_1 and S_2 can be computed as the dot product of the embeddings based on positive and negative eigenvalues respectively. Besides preserving features corresponding to both parts of the eigenvalue spectrum, SimEc can also be used to directly predict these two similarity matrices simultaneously. This can either be done by computing a new similarity matrix as $S_1 + S_2$ (Fig. 5 bottom), or by stacking the two matrices, thereby creating a tensor $\in \mathbb{R}^{m\times m\times 2}$. To preserve the information present in both similarity matrices to an equal extent, S_1 and S_2 first have to be normalized by their respective largest eigenvalue, as SimEc generally learn embeddings based on the overall largest eigenvalues. Unsurprisingly, the mean squared error between either S_1 or S_2 and $\hat{S}$ computed with a SimEc trained to approximate $S_1 + S_2$ is worse than that of a SimEc trained specifically to approximate either S_1 or S_2 alone (Fig. 7). The dot product of the embedding vectors $YY^\top$ of a SimEc trained to approximate the tensor containing the stacked matrices S_1 and S_2 also results in an error comparable to that of the $S_1 + S_2$ SimEc, because a single embedding contains the information about both similarity matrices here as well. However, the prediction of the individual similarity matrices in the tensor as YW_l yields errors as low as the prediction of the SimEc trained to approximate only one of the matrices, because the last dimension of the tensor W_l contains information specific to either one of the similarity matrices.

4. Discussion

Representing intrinsically complex structured data is an ubiquitous challenge in machine learning. While spectral methods such as kernel PCA provide optimal similarity preserving embeddings by computing the eigendecomposition of a similarity matrix, they are unable to produce OOS solutions for new test samples if their similarity to the original training examples can not be computed. Neural network based methods provide a mapping function from an original input feature space to the embedding space and can therefore also approximate the pairwise relations between new data points. However, existing methods were not designed to predict non-metric similarities or multiple pairwise relations simultaneously.

SimEc are a novel neural network architecture constructed for simultaneously learning a mapping from an original input feature space into a similarity preserving embedding space while factorizing a target matrix with pairwise relations. As we have demonstrated in multiple experiments, SimEc can provide OOS solutions even if the target similarities were obtained by an unknown process such as human ratings, they can efficiently handle missing values in the target matrix, and in addition they are able to predict non-metric similarities as well as multiple similarities at once.

While so far we mainly studied SimEcs based on fairly simple feed-forward neural networks, it appears promising to consider also deeper NN and more elaborate architectures, such as CNNs, for the initial mapping step to the embedding space. In this manner, hierarchical structures in complex data could be better reflected. Note furthermore that prior knowledge as well as more general error functions could be employed to tailor the embedding to the given targets.

In this paper we focused on using SimEc to predict pairwise similarities, but further application scenarios involving other pairwise relations between data points should be explored. For example, it has already been shown that a variant of SimEcs, called *context encoders* (ConEc) [16] learn meaningful word embeddings by extending the word2vec model [13, 14] for words with multiple meanings as well as to create out-of-vocabulary embeddings. The SimEc framework could also improve recommender systems or drug-protein interaction predictions and be interesting for usage in the sciences e.g., psychophysics [48], human quality judgment experiments [49], or materials discovery [50, 51].

Furthermore, future work will aim to interpret the predictions made by SimEc using layer-wise relevance propagation [52–55].

Acknowledgements. We would like to thank Antje Relitz, Maximilian Alber, and Christoph Hartmann for their helpful comments on earlier versions of this manuscript. Franziska Horn acknowledges funding from the Elsa-Neumann scholarship from the universities of Berlin. This work was supported by the Federal Ministry of Education and Research (BMBF) for the Berlin Big Data Center BBDC (No. 01IS14013A). Additional support was provided by the BK21 program funded by the Korean National Research Foundation Grant (No. 2012–005741) and the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (No. 2017–0–00451).

References

[1] C.M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[3] T. Hofmann and J. M. Buhmann, "Pairwise data clustering by deterministic annealing," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1), 1–14, 1997.

[4] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[5] L.v.d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research* 9, 2579– 2605, 2008.

[6] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, 10 (5), 1299–1319, 1998.

[7] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science* 290 (5500), 2319–2323, 2000.

[8] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science* 290 (5500), 2323–2326, 2000.

[9] F. Horn, "Interactive exploration and discovery of scientific publications with pubvis," *arXiv preprint arXiv:1706.08094*, 2017.

[10] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel pca and de-noising in feature spaces," in *Advances in Neural Information Processing Systems*, 536– 542, 1999.

[11] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks* 12 (2), 181–201, 2001.

[12] B. Schölkopf, S. Mika, C. J. Burges, Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Transactions on Neural Networks*, 10 (5), 1000–1017, 1999.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 3111–3119, 2013.

[15] Z. S. Harris, "Distributional structure," *Word* 10 (2–3), pp. 146–162, 1954.

[16] F. Horn, "Context encoders as a simple but powerful extension of word2vec," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 10–14, Association for Computational Linguistics, 2017.

[17] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems*, 2177–2185, 2014.

[18] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research* 12, pp. 2493–2537, 2011.

[19] Q.V. Le and T. Mikolov, "Distributed representations of sentences and documents," *arXiv preprint arXiv:1405.4053*, 2014.

[20] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of the 48th annual meeting of the Association for Computational Linguistics*, 384–394, Association for Computational Linguistics, 2010.

[21] M. Gönen, "Predicting drug–target interactions from chemical and genomic kernels using bayesian matrix factorization," *Bioinformatics*, 28 (18), 2304–2310, 2012.

[22] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer* 42 (8), 2009.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," tech. rep., Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[24] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *26th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6, IEEE, 2016.

[25] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd International Conference on World Wide Web*, 37–48, ACM, 2013.

[26] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.
[27] F. Chollet et al., "Keras." https://keras.io, 2015.
[28] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 263–272, IEEE, 2008.
[29] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology* 15 (3), 267–273, 1982.
[30] A. Cichocki, "Neural network for singular value decomposition," *Electronics Letters* 28 (8), 784–786, 1992.
[31] A. Cichocki and R. Unbehauen, "Neural networks for computing eigenvalues and eigenvectors," *Biological Cybernetics* 68 (2), 155–164, 1992.
[32] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer- Verlag New York, Inc., 1995.
[33] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, pp. 1177–1184, 2007.
[34] M. Alber, P.-J. Kindermans, K. Schütt, K.-R. Müller, and F. Sha, "An empirical study on the properties of random bases for kernel methods," in *Advances in Neural Information Processing Systems*, pp. 2760–2771, 2017.
[35] Y. Bengio, J.-f. Paiement, Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, "Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering," in *Advances in Neural Information Processing Systems*, 177–184, 2004.
[36] J. Mao and A. K. Jain, "Artificial neural networks for feature extraction and multivariate data projection," *IEEE Transactions on Neural Networks* 6 (2), 296–317, 1995.
[37] Y. W. Teh and S. T. Roweis, "Automatic alignment of local representations," in *Advances in Neural Information Processing Systems*, 865–872, 2003.
[38] M. A. Carreira-Perpinán and M. Vladymyrov, "A fast, universal algorithm to learn parametric nonlinear embeddings," in *Advances in Neural Information Processing Systems*, 253–261, 2015.
[39] L. van der Maaten, "Learning a parametric embedding by preserving local structure," in *International Conference on Artificial Intelligence and Statistics*, 384–391, 2009.
[40] K. Bunte, M. Biehl, and B. Hammer, "A general framework for dimensionality-reducing data visualization mapping," *Neural Computation* 24 (3), 771–804, 2012.
[41] D. Lowe and M. Tipping, "Feed-forward neural networks and topographic mappings for exploratory data analysis," *Neural Computing & Applications* 4 (2), 83–95, 1996.
[42] M. Kampffmeyer, S. Løkse, F. M. Bianchi, R. Jenssen, and L. Livi, "Deep kernelized autoencoders," in *Scandinavian Conference on Image Analysis*, 419–430, Springer, 2017.
[43] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, 2, 1735–1742, IEEE, 2006.
[44] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22nd ACM international conference on information & knowledge management*, pp. 2333–2338, ACM, 2013.
[45] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "Starspace: Embed all the things!," *arXiv preprint arXiv:1709.03856*, 2017.
[46] J. Laub and K.-R. Müller, "Feature discovery in non-metric pairwise data," *Journal of Machine Learning Research* 5, 801–818, 2004.
[47] F. Horn. https://github.com/cod3licious/simec/blob/ master/experiments_ paper.ipynb.
[48] J. Laub, K.-R. Müller, F.A. Wichmann, and J.H. Macke, "Inducing metric violations in human similarity judgements," in *Advances in Neural Information Processing Systems*, 777–784, 2007.
[49] S. Bosse, D. Maniry, K.-R. Müller, T. Wiegand, and W. Samek, "Deep neural networks for no-reference and full-reference image quality assessment," *IEEE Transactions on Image Processing*, 27 (1), 206–219, 2018.
[50] K.T. Schütt, F. Arbabzadah, S. Chmiela, K.-R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature communications* 8, 13890, 2017.
[51] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "Schnet–a deep learning architecture for molecules and materials," *The Journal of Chemical Physics* 148 (24), 241722, 2018.
[52] L. Arras, F. Horn, G. Montavon, K.-R. Müller, and W. Samek, "what is relevant in a text document?": An interpretable machine learning approach," *PLOS ONE* 12 (8), e0181142, 2017.
[53] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, andW. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, 10 (7), e0130140, 2015.
[54] P.-J. Kindermans, K.T. Schütt, M. Alber, K.-R. Müller, and S. Dähne, "Patternnet and patternlrp–improving the interpretability of neural networks," *arXiv preprint arXiv:1705.05598*, 2017.
[55] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition* 65, 211– 222, 2017.