Representation Interpretation with Spatial Encoding and Multimodal Analytics

Ninghao Liu, Mengnan Du, Xia Hu
Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA, 77843
{nhliu43,dumengnan,xiahu}@tamu.edu

ABSTRACT

Representation learning models map data instances into a low-dimensional vector space, thus facilitating the deployment of subsequent models such as classification and clustering models, or the implementation of downstream applications such as recommendation and anomaly detection. However, the outcome of representation learning is difficult to be directly understood by users, since each dimension of the latent space may not have any specific meaning. Understanding representation learning could be beneficial to many applications. For example, in recommender systems, knowing why a user instance is mapped to a certain position in the latent space may unveil the user's interests and profile. In this paper, we propose an interpretation framework to understand and describe how representation vectors distribute in the latent space. Specifically, we design a coding scheme to transform representation instances into spatial codes to indicate their locations in the latent space. Following that, a multimodal autoencoder is built for generating the description of a representation instance given its spatial codes. The coding scheme enables indication of position with different granularity. The incorporation of autoencoder makes the framework capable of dealing with different types of data. Several metrics are designed to evaluate interpretation results. Experiments under various application scenarios and different representation learning models are conducted to demonstrate the flexibility and effectiveness of the proposed framework.

KEYWORDS

Interpretation; Representation Learning; Recommender Systems

ACM Reference Format:
Ninghao Liu, Mengnan Du, Xia Hu. 2019. Representation Interpretation with Spatial Encoding and Multimodal Analytics. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3290960

1 INTRODUCTION

Representation learning has a fundamental impact on the performance of downstream machine learning models, especially when processing raw data inputs such as images [41], texts [2, 26] and networks [29, 35]. Representation learning is sometimes explicitly pursued by models such as autoencoders [37] or embedding models [26, 29], while it may also implicitly make an effect in many applications such as multimodal learning [14], recommender systems [19] and anomaly detection [3]. Data points with similar properties or semantic meanings are mapped close to each other in the representation space, where the semantics depend on the dataset and the task objective.

Although representation learning lays the foundation of the success of various machine learning tasks, its interpretability has not been fully explored. The opacity of representation space originates from the fact that each of its dimensions may not have any specific meaning. Current techniques for evaluating the quality of representation learning focus on the performance of subsequent tasks such as classification accuracy and recommendation precision. However, how representation vectors distribute in the latent space, which directly affects the results of subsequent tasks, is usually opaque to users. The interpretation of representation result could benefit the application of machine learning models in several ways. On one hand, since representation learning encodes various features or different types of information into the same space, we are interested in which information sources play an important role and which ones are actually useless in the learning process. It may help us evaluate the effectiveness of representation result. On the other hand, interpretation could help justify the decisions made by machine learning models. For example, in recommender systems, especially those built upon deep models [11, 42] that are usually regarded as black boxes, we may want to know why certain products are recommended to the target user.

Understanding the outcome of representation learning is a challenging problem due to several factors. First, most of the existing interpretation methods, which mainly focus on understanding classification models, cannot be directly applied to our problem [4, 15, 30], since there is no notion of classification boundaries or labels. Some recent work tries to comprehend the embedding space [23], but interpretation is directly attached to clusters in the space and is not available for individual instances. Second, a flexible interpretation framework is desirable, so that its inner modules can easily adapt to different types of data. Third, visualization techniques [25, 34] are heavily relied on by users to understand how instances distribute in the latent space. However, it is difficult to initiate detailed and objective analysis based on visualization, let alone that the accuracy of visualization may not even be guaranteed.

To overcome the challenges above, in this paper, we propose a new framework for understanding the outcome of representation learning. The framework is post-hoc, so it does not rely on specific embedding method that outputs the representation instances.

WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5940-5/19/02…$15.00
https://doi.org/10.1145/3289600.3290960

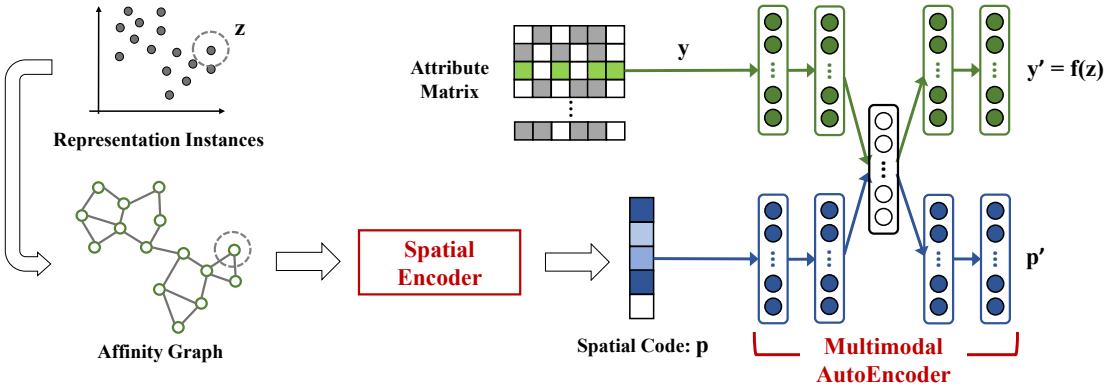

Figure 1: The proposed interpretation approach, whose core module is a multimodal autoencoder. One modality is spatial code, and the other is attribute information.

Different from existing interpretation methods that target on classification models, or provide only cluster-level explanation for unsupervised models, the proposed framework can describe the representation space regions with flexible granularity. As the corpus for generating intuitive description to instances, an attribute matrix is introduced, which could either be set as the input data to the representation learning model (e.g., word tokens in document embedding tasks) or constructed from side information in the application (e.g., tags in recommender systems). Specifically, our framework first transforms representation instances into spatial codes to indicate where the instances locate in the representation space. Then, we train a multimodal autoencoder to establish the relation between the spatial codes and attribute matrix. After that, the description of a target representation instance can be generated by only feeding its spatial code into the autoencoder. The architecture of the autoencoder can be determined based on the data type of attributes. The major contributions of this paper are as follows:

- We design an interpretation framework to understand the outcome of representation learning. The framework can be applied to different types of representation learning methods.
- We develop a coding module to represent the spatial position of representation instances with varying granularity. Also, the multimodal autoencoder (i.e., the interpretation generator) can adapt to different types of data.
- We conduct experiments on various machine learning applications and different representation learning models, to demonstrate the flexibility and effectiveness of the proposed framework.

2 PROBLEM FORMULATION

2.1 Notations

We use boldface uppercase characters (e.g., $\mathbf{A}$) to denote matrices, boldface lowercase characters (e.g., $\mathbf{z}$) as vectors, calligraphic characters (e.g., $\mathcal{V}$) as sets, and normal letters (e.g., i, K) as scalars. The size of a set $\mathcal{V}$ is denoted as $|\mathcal{V}|$. We represent the i-th row of $\mathbf{A}$ as $\mathbf{A}(i,:)$, the j-th column as $\mathbf{A}(:,j)$, and the (i,j) entry $\mathbf{A}(i,j)$. The transpose of a vector $\mathbf{z}$ or a matrix $\mathbf{A}$ is denoted as $\mathbf{z}^T$ or $\mathbf{A}^T$ (instead of $\mathbf{z}'$ or $\mathbf{A}'$ which is used to refer to different vector or matrix). $\mathbf{z}(j)$ denotes the j-th entry of the vector. Following the rules above, let $\mathbf{X} \in \mathbb{R}^{N\times K}$ denote the input processed by representation learning, where there are N objects and each object is associated with K features. The representation learning generates $\mathbf{Z} \in \mathbb{R}^{N\times D}$, where D is the dimension of the latent space, and $\mathbf{Z}(i,:) \in \mathbb{R}^D$ denotes the representation instance of the i-th node. We also use $\mathbf{z} \in \mathbb{R}^D$ to denote a representation instance. In this paper, we use representation *vectors* and representation *instances* interchangeably.

In this paper, we also introduce an *attribute matrix* denoted as $\mathbf{Y} \in \mathbb{R}^{N\times M}$ as the corpus for interpretation generation. The construction of $\mathbf{Y}$ depends on the application scenarios. In some cases, $\mathbf{Y}$ is extracted from $\mathbf{X}$ or even equals to $\mathbf{X}$. For examples, in text mining, each column of $\mathbf{Y}$ could represent the appearance of a word token. In network analysis, $\mathbf{Y}$ could be the side information that describes the properties of network nodes. In some other cases, $\mathbf{Y}$ is disjoint from $\mathbf{X}$. For example, in recommender systems, $\mathbf{Y}$ could be the summary of customer interests, review contents or visual appearance of products. The overall principle for constructing $\mathbf{Y}$ is that each attribute in $\mathbf{Y}$ should be directly comprehensible to humans.

2.2 Interpretation Framework Overview

The goal of interpretation for representation instances is to build a mapping function $f : \mathbb{R}^D \to \mathbb{R}^M$, where the input representation vector $\mathbf{z}$ is mapped to characteristic attributes $\mathbf{y}' \in \mathbb{R}^M$. The resultant $\mathbf{y}'$ contains the descriptive properties of the given input $\mathbf{z}$. Here we use $\mathbf{y}'$, instead of $\mathbf{y}$, to emphasize that they have different meanings although they are in the same space.

In this work, we build the interpretation mapping function f by resorting to *multimodal analytics*. For each object in a dataset, such as a customer in recommender systems or a node in network embedding, it is involved with two information modalities: the position information contained in its embedding vector $\mathbf{z}$, and the characteristic information expressed by the attribute vector $\mathbf{y}$. Some elements in $\mathbf{y}$ are relevant with $\mathbf{z}$, while others are just noises. Thus, after establishing the correlation between the two modalities, with $\mathbf{Z}$ and $\mathbf{Y}$ as training samples, the interpretation $\mathbf{y}'$ can be obtained through multimodal inference given the input query $\mathbf{z}$.

The overall interpretation approach is shown in Figure 1. The interpretation function f is the composite of two modules, i.e., $f(\mathbf{z}) = (g \circ h)(\mathbf{z}) = g(h(\mathbf{z}))$. Here $h : \mathbb{R}^D \to \mathbb{R}^{C_P}$ is a spatial

encoder which transforms representation vector $\mathbf{z}$ into a vector $\mathbf{p} \in \mathbb{R}^{C_p}$ called *spatial codes*, while $g : \mathbb{R}^{C_p} \rightarrow \mathbb{R}^M$ reconstructs attributes $\mathbf{y}'$ from spatial codes. The role of spatial codes is to intuitively indicate the location of a representation vector in the latent space. After that, we design g as part of a multimodal autoencoder which handles the correlation between position information and attribute information. The details of each module are introduced in the sections below.

3 SPATIAL ENCODING FOR REPRESENTATION INSTANCES

In this section, we introduce details of building the spatial encoder. The role of the spatial encoder is to assign representation instances with spatial codes specifying which clusters the instances belong to. In this way, we can intuitively express the relative location of instances in the latent space.

3.1 Affinity Graph based on Representations

The relations between representation vectors are preserved through their mutual similarities in the latent space. For mining such relations, given the representation vectors $\mathbf{Z}$, we first build an affinity graph $\mathcal{G}$ to efficiently store their similarities in the latent space. Let $\mathbf{A}$ denote the affinity matrix of the graph, the link weight between a pair of nodes is defined as:

$$\mathbf{A}_{i,j} = \begin{cases} e_{i,j}, & \text{if } j \in nbrs(i) \text{ or } i \in nbrs(j) \\ 0, & \text{otherwise} \end{cases}, \tag{1}$$

and

$$e_{i,j} = 1 - \frac{\mathbf{Z}(i,:)\mathbf{Z}(j,:)^T}{\|\mathbf{Z}(i,:)\|\|\mathbf{Z}(j,:)\|}, \tag{2}$$

where $nbrs(i)$ denotes the set of nearest neighbors of node i. The number of neighbors can be set as $|nbrs(\cdot)| = b\lceil \log_2 N \rceil$ according to [39] and b is a constant integer. $|nbrs(i)|$ is much smaller than the number of nodes in $\mathcal{G}$. In this way, we only maintain a sparse affinity matrix instead of computing the weights of all node pairs which could be expensive for both computation and storage [25, 34]. The distances between representation vectors, i.e., the relations between the corresponding real-world instances, are recovered through the link information in $\mathcal{G}$.

3.2 Encoding Representation Vectors Under Multiple Resolutions

After obtaining the affinity graph $\mathcal{G}$ constructed from representation vectors, we will assign each graph node into some clusters to denote its relative position in the graph and, hence, in the representation space. The clustering is implemented on $\mathcal{G}$, rather than directly on $\mathbf{Z}$, because it can capture the nonlinear manifolds in data [46]. We choose symmetric nonnegative matrix factorization (SNMF) as the basic model to find clusters contained in $\mathcal{G}$ [40]:

$$\min_{\mathbf{W} \geq 0} \|\mathbf{A} - \mathbf{W}\mathbf{W}^T\|_F^2 + \gamma\|\mathbf{W}\|_F^2, \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{N \times C}$, C is the hyperparameter deciding how many clusters to define, and γ controls the effect of the normalization term. The matrix factorization problem above can also be seen as the Kernel K-means clustering of $\mathbf{Z}$ with the orthogonality of $\mathbf{W}$ relaxed [6], where the kernel matrix is computed based on Equation 1 and Equation 2. Each entry $\mathbf{W}(n,c)$ indicates the strength that node n is associated with cluster c. The relative position of node n in the representation space is thus encoded in $\hat{\mathbf{W}}(n,:)$, where $\hat{\mathbf{W}}$ denotes the normalized factor matrix so that $\sum_{c=1}^{C} \hat{\mathbf{W}}(n,c) = 1$.

There are several limitations of simply using $\mathbf{W}$ to encode the position of nodes. First, it is difficult to have a good choice for the value of C. Although we may resort to validation techniques [36] to estimate the number of clusters contained in $\mathcal{G}$, it is inefficient when the data size is large. Second, it is not flexible enough to only use flat clustering to encode the position of an instance. The resultant clusters are independent of each other. For a large cluster, it is difficult to further discriminate between the attributes of different nodes inside it. For a small cluster, it is harder to accurately identify its signaling characteristics due to noises and lack of data. Third, implicit hierarchical structures naturally exist in many datasets [43], such as in recommender systems and publication networks, but such structures cannot be captured through flat clustering. We wonder if there is a way to avoid the difficulty of determining a single optimal C value, and have an effective and flexible coding scheme to represent the positions of instances.

3.2.1 Hierarchical Structure Generation. To address the difficulties above, we propose a hierarchical encoding method to represent the position of vectors in latent spaces. The $\mathbf{W} \in \mathbb{R}^{N \times C_1}$ obtained from flat clustering, where C_1 equals to C, indicates the affiliation of N nodes to C_1 clusters. Since $\mathbf{W}$ is also nonnegative, we can further decompose it into two nonnegative matrices $\mathbf{W}_2' \in \mathbb{R}^{N \times C_2}$ and $\mathbf{W}_1 \in \mathbb{R}^{C_2 \times C_1}$ to get a 2-layer hierarchical cluster structure, i.e., $\mathbf{W} \approx \mathbf{W}_2'\mathbf{W}_1$. Here C_2 is the number of clusters in the 2-nd layer, $\mathbf{W}_2'$ indicates the affiliation of N nodes to 2-nd layer clusters, and $\mathbf{W}_1$ indicates the affiliation relation between C_1 clusters in the 1-st layer and C_2 clusters in the 2-nd layer. Similarly, since $\mathbf{W}_2'$ is also nonnegative, it can be further decomposed into $\mathbf{W}_3' \in \mathbb{R}^{N \times C_3}$ and $\mathbf{W}_2 \in \mathbb{R}^{C_3 \times C_2}$ to get the 3-layer hierarchical cluster structure. This process can be generalized to a L-layer hierarchical structure:

$$\mathbf{W} \approx \mathbf{W}_L\mathbf{W}_{L-1}...\mathbf{W}_2\mathbf{W}_1, \tag{4}$$

where $\mathbf{W}_L$ is a $N \times C_L$ matrix, and $\mathbf{W}_l$ is a $C_{l+1} \times C_l$ matrix. Here $\mathbf{W}_L$ is not written as $\mathbf{W}_L'$, because it will no longer be factorized. We let $C_l < C_p$ for $l < p$, which means higher layers are of finer granularity. The overall loss function to solve the hierarchical clustering problem discussed above is formulated as:

$$\begin{aligned} \min_{\mathbf{W}_1,\ldots,\mathbf{W}_L} \ & \|\mathbf{A} - \mathbf{W}_L...\mathbf{W}_1\mathbf{W}_1^T...\mathbf{W}_L^T\|_F^2 + 2\gamma\Big(\sum_{l=1}^{L}\|\mathbf{W}_l\|_F^2\Big) \\ \text{s.t.} \ & \mathbf{W}_l \geq 0,\ l \in \{1,2,...,L\}. \end{aligned} \tag{5}$$

In this case, larger C_l's in shallow layers generates clusters with finer granularity, while smaller C_l's in deeper layers regulate the relation among small clusters. We are thus able to specify positions in the latent space with enough flexibility through the hierarchical structure. In our experiments, we choose the value of C_L to be larger than the estimated number of clusters in $\mathbf{Z}$, while we set C_1 to be much smaller. An illustration of the hierarchical structure is in Figure 2, where there are three cluster layers and a bottom layer.

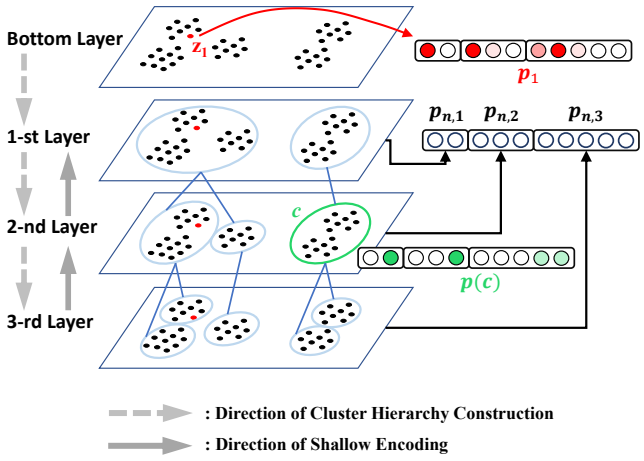

Figure 2: An example of spatial codes for representation instances. There are three layers in the spatial encoder.

3.2.2 Spatial Encoding for Existing Instances. After obtaining the hierarchical cluster structure, we are able to represent the spatial position of an instance in the latent space in an effective way. An instance n has different cluster assignments on different layers of varying granularity. Suppose we let a row vector $\mathbf{p}_{n,l} \in \mathbb{R}^{C_l}$ denote the cluster assignment of instance n in the l-th layer, then its overall spatial code $\mathbf{p}_n$ is defined as the concatenation of its cluster assignments on different layers, i.e., $\mathbf{p}_n = [\hat{\mathbf{p}}_{n,1}, \hat{\mathbf{p}}_{n,2}, ..., \hat{\mathbf{p}}_{n,L-1}, \hat{\mathbf{p}}_{n,L}]$, as shown in Figure 2. Here $\hat{\mathbf{p}}$ denotes the probability vector after normalizing $\mathbf{p}$ so that the entries sum to 1. Specifically, $\mathbf{p}_{n,l}$ can be computed as:

$$\mathbf{p}_{n,l} = \mathbf{e}_n \mathbf{W}_L \mathbf{W}_{L-1} ... \mathbf{W}_l, \tag{6}$$

where $\mathbf{e}_n \in \mathbb{R}^N$ is a one-hot row vector in which $\mathbf{e}_n(i) = 1$ for $i = n$ and $\mathbf{e}_n(i) = 0$ for $i \neq n$, $1 \leq i \leq N$.

3.2.3 Spatial Encoding for New Instances. Given a new representation vector $\mathbf{z}_\nu$ beyond the existing dataset, we first estimate its linkage $\mathbf{a}_\nu \in \mathbb{R}^N$ to the existing N nodes in $\mathcal{G}$ according to Equation 1 and Equation 2. Then, the spatial code of ν is $\mathbf{p}_\nu = [\hat{\mathbf{p}}_{\nu,1}, \hat{\mathbf{p}}_{\nu,2}, ..., \hat{\mathbf{p}}_{\nu,L-1}, \hat{\mathbf{p}}_{\nu,L}]$, where

$$\mathbf{p}_{\nu,l} = \mathbf{a}_\nu \mathbf{W}_L \mathbf{W}_{L-1} ... \mathbf{W}_l. \tag{7}$$

In this way, we can investigate a wider range of continuous regions in the latent space, out of the exact locations of representation vectors in $\mathbf{Z}$. If the number of new instances is large and the distribution of representation instances could be affected, we may consider dynamically updating the instances, but this is beyond the discussion of this work.

3.3 Optimization for Hierarchical SNMF

It is challenging to directly solve the hierarchical SNMF, due to (1) the correlation among factor matrices, and (2) the symmetric duplicate of each factor matrix in the loss function. The common way to address the first challenge is to apply iterative optimization algorithms and update factor matrices alternatively [7], i.e., updating one factor matrix at each iteration with other matrices fixed.

Algorithm 1: Spatial Encoding for Representation Vectors

Input: $\mathbf{Z}$, L, $\{C_1, ..., C_L\}$, β_0, r_β
Output: Spatial code $\mathbf{p}_n$ for each instance n

```
Build the affinity matrix A based on Equation 1 and Equation 2.
Randomly initialize {W_1, W_2, ..., W_L}, where W_L ∈ R^{N×C_L} and W_l ∈ R^{C_{l+1}×C_l}.
for l = L : −1 : 1 do
    Pre-train W_l according to min_{W_l} ||A^l − W_l W_l^T||_F^2, where A^l = W_{l+1}^T W_{l+1} for l ≠ L and A^L = A.
Initialize {H_1, H_2, ..., H_L} as H_l = W_l, 1 ≤ l ≤ L.
β = β_0.
while not converged do
    for l = 1 : L do
        Update W_l according to Equation 10;
    for l = 1 : L do
        Update H_l according to Equation 11;
    β ← β · r_β.
Obtain the spatial codes for existing instances or new instances, respectively according to Equation 6 or Equation 7.
```

To address the second challenge, we reformulate Equation 5 as:

$$\begin{aligned} \min_{\mathbf{W}_l, \mathbf{H}_l, l \in [1,L]} \; & \|\mathbf{A} - \mathbf{W}_L ... \mathbf{W}_1 \mathbf{H}_1^T ... \mathbf{H}_L^T\|_F^2 + \beta \Big(\sum_{l=1}^{L} \|\mathbf{W}_l - \mathbf{H}_l\|_F^2\Big) \\ & + \gamma \Big(\sum_{l=1}^{L} \|\mathbf{W}_l\|_F^2 + \|\mathbf{H}_l\|_F^2\Big), \\ \text{s.t.} \quad & \mathbf{W}_l, \mathbf{H}_l \geq 0, \; l \in \{1, 2, ..., L\}, \end{aligned} \tag{8}$$

where $\mathbf{H}_l$'s are introduced to decouple factor matrices from their duplicates, and β controls the tradeoff between NMF error and matrix differences [20]. To guarantee $\mathbf{W}_l$ and $\mathbf{H}_l$ to be close to each other, we gradually increase the value of β as iterations progress until convergence.

Suppose $\mathbf{W}_l$ is to be updated at the current iteration, then the loss function for updating $\mathbf{W}_l$ can be rewritten as:

$$\min_{\mathbf{W}_l \geq 0} \|\mathbf{A} - \mathbf{B}_l \mathbf{W}_l \mathbf{F}_l\|_F^2 + \beta \|\mathbf{W}_l - \mathbf{H}_l\|_F^2 + \gamma \|\mathbf{W}_l\|_F^2, \tag{9}$$

where $\mathbf{B}_l = \mathbf{W}_L ... \mathbf{W}_{l+1}$ if $l \neq L$ and $\mathbf{B}_l = \mathbf{I} \in \mathbb{R}^{N \times N}$ if $l = L$, and $\mathbf{F}_l = \mathbf{W}_{l-1} ... \mathbf{W}_1 \mathbf{H}_1^T ... \mathbf{H}_L^T$ if $l \neq 1$ and $\mathbf{F}_l = \mathbf{H}_1^T ... \mathbf{H}_L^T$ if $l = 1$. By utilizing the Lagrangian function and auxiliary function method [7, 22, 43], the update rule for $\mathbf{W}_l$ is

$$\mathbf{W}_l(i,j) \leftarrow \mathbf{W}_l(i,j) \frac{[\mathbf{B}_l^T \mathbf{A} \mathbf{F}_l^T + \beta \mathbf{H}_l](i,j)}{[\mathbf{B}_l^T \mathbf{B}_l \mathbf{W}_l \mathbf{F}_l \mathbf{F}_l^T + \beta \mathbf{W}_l + \gamma \mathbf{W}_l](i,j)}. \tag{10}$$

Similarly, the update rule for $\mathbf{H}_l$ is as below:

$$\mathbf{H}_l(i,j) \leftarrow \mathbf{H}_l(i,j) \frac{[\mathbf{F}_l' \mathbf{A}^T \mathbf{B}_l' + \beta \mathbf{W}_l](i,j)}{[\mathbf{F}_l' \mathbf{F}_l'^T \mathbf{H}_l \mathbf{B}_l'^T \mathbf{B}_l' + \beta \mathbf{H}_l + \gamma \mathbf{H}_l](i,j)}. \tag{11}$$

where $\mathbf{B}_l' = \mathbf{W}_L ... \mathbf{W}_1 \mathbf{H}_1^T ... \mathbf{H}_{l-1}^T$ if $l \neq 1$ and $\mathbf{B}_l' = \mathbf{W}_L ... \mathbf{W}_1$ if $l = 1$, and $\mathbf{F}_l' = \mathbf{H}_{l+1}^T ... \mathbf{H}_L^T$ if $l \neq L$ and $\mathbf{F}_l' = \mathbf{I}$ if $l = L$.

The overall process of spatial encoding is presented in Algorithm 1. First, given the representation matrix $\mathbf{Z}$, we construct the sparse affinity matrix $\mathbf{A}$ to store the distances between representation vectors. Then, we pre-train the factor matrices $\mathbf{W}_l$, and

initialize $\mathbf{H}_l$ as copies of $\mathbf{W}_l$. After that, we iteratively update $\mathbf{W}_l$ and $\mathbf{H}_l$ following the multiplicative updating formulas. The value of β increases as iterations proceed, in order to force $\mathbf{W}_l$ and $\mathbf{H}_l$ close to each other. Finally, the spatial codes of representation vectors can be computed through mapping to established clusters.

4 INTERPRETATION GENERATION MODULE

In the previous section, we have introduced the spatial codes $\mathbf{p}$ to intuitively indicate where an instance or a region locates in the representation space. In this section, we will introduce how to generate interpretation given $\mathbf{p}$ through constructing a multimodal denoising autoencoder.

4.1 Multimodal Autoencoder

The structure of the multimodal autoencoder used in this work is shown in the right part of Figure 1. The goal of the autoencoder is to establish the correlation between the spatial location of instances and their attributes. Let g^E denote the encoder layers, while g_y^D and g_p^D be the decoder layers for recovering attributes and spatial codes, respectively. The objective function for data recovery can be expressed as:

$$\min_{g^E, g_y^D, g_p^D} \frac{1}{N}\sum_{n=1}^{N}\frac{1}{J}\sum_{j=1}^{J}\Big(l_y(\mathbf{y}_n, g_y^D \circ g^E(\tilde{\mathbf{p}}_n^j, \tilde{\mathbf{y}}_n^j)) + l_p(\mathbf{p}_n, g_p^D \circ g^E(\tilde{\mathbf{p}}_n^j, \tilde{\mathbf{y}}_n^j))\Big), \tag{12}$$

which minimizes the difference between clean data samples $\{\mathbf{y}_n, \mathbf{p}_n\}$ and reconstructed data output from $g_y^D \circ g^E$ and $g_p^D \circ g^E$. Here we denote $\mathbf{y}_n$ as $\mathbf{Y}(n,:)$ for conciseness. $\tilde{\mathbf{p}}_n^j$ is the j-th corruption of the spatial code of the n-th instance, and $\tilde{\mathbf{y}}_n^j$ is the j-th corruption of its attributes. $l_y(\cdot,\cdot)$ and $l_p(\cdot,\cdot)$ denote the reconstruction error with respect to attributes and spatial codes, respectively.

After training the multimodal autoencoder, given an instance $\mathbf{z}$ whose characteristics we hope to explore, we first obtain its corresponding spatial code $\mathbf{p}$. Then, the interpretation for $\mathbf{z}$ is:

$$f(\mathbf{z}) = g \circ h(\mathbf{z}) = g(\mathbf{p}) = g_y^D \circ g^E(\mathbf{p}, \mathbf{0}), \tag{13}$$

where we set the attribute modality to be absent but infer the attributes value based on $\mathbf{p}$. In this case, $l_p(\cdot,\cdot)$ in the autoencoder training stage can be seen as to project spatial codes to the low-dimensional manifold of the training data.

4.2 Autoencoder Training Procedure

Inspired by multimodal learning [28] and denoising autoencoders [38], the training procedure for our multimodal autoencoder involves two aspects. First, we use training samples that set one of the input modality with zero values (e.g., $\tilde{\mathbf{y}}_n^j = \mathbf{0}$) and keep the other input modality as original, but still require the autoencoder to reconstruct both modalities. Second, we further corrupt the training samples on the input side. Specifically, we apply Gaussian noise corruption to obtain $\tilde{\mathbf{y}}$, and apply salt-and-pepper corruption [38] to obtain $\tilde{\mathbf{p}}$. Gaussian noise is a very common noise model for real valued inputs. For salt-and-pepper corruption, it means setting a fraction of entries in $\mathbf{p}$ randomly to their minimum or maximum possible value. Since each entry of $\mathbf{p}$ means the probability of belonging to a certain cluster, the minimum and maximum value are 0 and 1, respectively. Larger entries in $\mathbf{p}$ are more likely to be set as 1.

When adopting the "denoising" strategy, we apply salt-and-pepper noise on spatial codes because it naturally aligns with our goal of interpretation. A spatial code corrupted by salt-and-pepper noise is actually a *cluster indicator*. For example, suppose we have a two-layer spatial code $\mathbf{p} = [0.8, 0.2, 0.7, 0.2, 0, 0.1]$, where there are two clusters on the 1-st layer and four clusters on the 2-nd layer, some possible corrupted codes could be $\tilde{\mathbf{p}} = [1, 0, 1, 0, 0, 0]$ with high probability or $\tilde{\mathbf{p}} = [0, 1, 0, 0, 0, 1]$ with low probability. When feeding these samples to the autoencoder, instead of establishing the relation solely between the instance $\mathbf{p}$ to its attributes, we are forcing the autoencoder to learn the relation between the clusters and the attributes of samples inside them. Till now, we have introduced the whole framework of interpreting the outcome of representation learning, including coding the spatial positions of representation instances, as well as building a multimodal autoencoder to generate interpretation given spatial codes.

Dataset	N	M
20NG	11,314	1,200
Flickr	7,575	69
MovieLens	6,040	1,128

Table 1: Statistics of datasets.

5 EXPERIMENTS

We evaluate the effectiveness and flexibility of the proposed interpretation framework based on different real-world datasets and different representation learning models. Two strategies are designed to quantitatively measure whether the generated interpretations are faithful to representation learning models.

5.1 Experimental Settings

5.1.1 Datasets. The datasets used in experiments, as well as the definition of attributes for each dataset, are summarized as below. The statistics of the datasets are in Table 1, where N is the number of instances and M denotes the dimension of attributes.

- **20NG**: The 20NewsGroup dataset contains a word-document matrix. Each document is seen as an instance. The attributes of documents are defined as the TF-IDF vectors constructed from the text corpus.
- **Flickr** [13]: A network dataset constructed based on links between online bloggers and their attributes. Bloggers are treated as nodes. Each node is associated with two types of attributes: (1) word attributes obtained from the posts generated by bloggers; (2) personal interests of bloggers. The two sets of attributes are concatenated for each node.
- **MovieLens** [10]: A movie rating dataset which has been widely used in evaluating collaborative filtering recommender systems. The dataset consists of a rating matrix and a tag information matrix. Ratings are made on the 5-star scale. Each user has at least 20 ratings . The attributes of each user $\mathbf{Y}(u,:)$ are constructed by utilizing his/her rating records combined with the tag information of movies. Specifically, $\mathbf{Y}(u,m) = \sum_v r(u,v) \cdot c(v,m)$, where v denotes a movie, m denotes a tag, $r(u,v)$ is the rating that user

Dataset	#neurons in g_y^D layers	#neurons in g_p^D layers
20NG	25 - 150 - 500 - M	25 - C_p
Flickr	10 - 25 - M	10 - C_p
MovieLens	35 - 150 - 400 - M	35 - C_p

Table 2: Neural network structure of decoder for each modality in the multimodal autoencoder used on different datasets.

u gives to movie v, and $c(v, m)$ measures how closely the tag m is associated with movie v.

The three datasets above have been applied in different domains, i.e., text analysis, network embedding and recommender systems, respectively. We will test our framework on these datasets to evaluate its effectiveness and versatility.

5.1.2 The Applied Representation Learning Models. For each dataset, we implement several representation learning models whose results are to be interpreted. For 20NG data, we use NMF [22] and Doc2VecC [2] to learn the representation of documents. The input fed into NMF is the TF-IDF matrix of size $N \times M$ built from the data corpus, where N is the number of documents and M is the vocabulary size. The input into Doc2VecC is the raw text. The latent space dimension is 50 for NMF and 100 for Doc2VecC. For Flickr data, we use LANE [13] and CMF [47] to learn the embedding of network nodes. Both models utilize links and attributes together to jointly learn the embeddings of nodes. The latent space dimension is set as 100 for LANE and 30 for CMF. For MovieLens, we use NMF-based collaborative filtering (NMF-CF) [19] and neural collaborative filtering (NeuCF) [11] to learn the representation of users. The input fed into the recommender systems is the user-movie rating matrix. The latent space dimension is set as 50 for NMF-CF and 16 for NeuCF. The details of tuning each model mainly follow the paper references and are omitted here due to page limit. After obtaining the representation vectors, we will feed them together with attributes into our interpretation method.

5.1.3 Details of Framework Implementation. We use two layers for all spatial encoders in experiments. The dimension of each layer is set as follows: $C_1 = 10$ and $C_2 = 25$ for interpreting NMF and Doc2VecC on 20NG data, $C_1 = 9$ and $C_2 = 15$ targeting LANE and CMF on Flickr data, $C_1 = 18$ and $C_2 = 35$ for interpreting NMF-CF and NeuCF on MovieLens data. We use variational autoencoders [17] as the basis for building the multimodal autoencoders. Fully connected networks are chosen as the architecture for both encoders and decoders. The structure of multimodal autoencoders is shown in Table 2, where $C_p = C_1 + C_2$ for the input layer of g_p^D. In this table, we only show the number of neurons in the layers of decoders g_p^D and g_y^D respectively, while the structure of encoder is symmetric to that of its corresponding decoder for each modality.

To have some intuitive sense of the effect of spatial encoding, we provide some example in Figure 3, using t-SNE [25] for 2D space visualization. The two scatter plots show the clustering result on the representation vectors obtained from Doc2VecC on 20NG data. Each color corresponds to a cluster. Each vector is assigned to the cluster corresponding to the largest entry in the spatial codes. There are two layers in the spatial codes, where $C_1 = 10$ for the left plot and $C_2 = 25$ for the right plot. We could observe from the figure that cluster assignments are in general consistent with

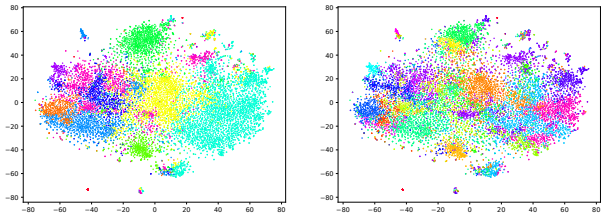

Figure 3: Visualization of spatial codes, with two hierarchies, on the representations of documents in 20NG dataset. Left: 10 clusters; Right: 25 clusters.

the distribution of representation vectors. The two sets of clusters mutually complement each other. Fine-grained spatial codes can discover more details of cluster structures, while coarse-grained spatial codes alleviate excessive cluster partitioning.

5.1.4 Baseline Methods. We will compare the proposed method with several other model-agnostic interpretation methods and some of its variants as baseline methods, regarding the interpretation faithfulness. The baseline methods are listed as below.

- **LIME** [30]: An interpretation model originally designed for explaining individual predictions of a given classification model. In our experiments, given a representation vector to be interpreted, we use LIME with LASSO to select important attributes as its interpretation, through comparing the neighborhoods of the target vector with some other distant clusters.
- **MTGL** [16]: A multitask classification model with regularization of tree-based group lasso. An affinity graph is also built for hierarchical clustering, where graph nodes are deterministically assigned to clusters organized in a hierarchy structure [23]. The resultant clusters are regarded as different classes in linear classification. The value of linear coefficients in the model is used as interpretation for different clusters. The interpretation of each node follows the interpretation of the cluster it belongs to.
- **FlatCoarse**: A variant of the proposed method, where flat clustering replaces the hierarchical solution. Specifically, the number of clusters is set as C_1, so the representation space is partitioned with coarse granularity.
- **FlatFine**: Another flat-clustering variant of the proposed model, where the number of clusters is set as C_L, so the representation space is partitioned with fine granularity.

5.2 Interpretation Evaluation Through Adversarial Perturbation

The goal of this experiment is to evaluate whether the interpretation generated by our framework can correctly identify the attributes that preserve the similarities contained in representation vectors. The core idea of evaluating interpretation correctness is to check the effectiveness of the *adversarial perturbation* initiated based on the interpretation. The motivation behind this evaluation method is that knowing how a model operates provides us the direction to "hack" it [5][18][24].

5.2.1 Interpretation Evaluation for Document Representation Learning. We first evaluate the interpretation accuracy for text representation learning. For continuous attributes such as TF-IDF values

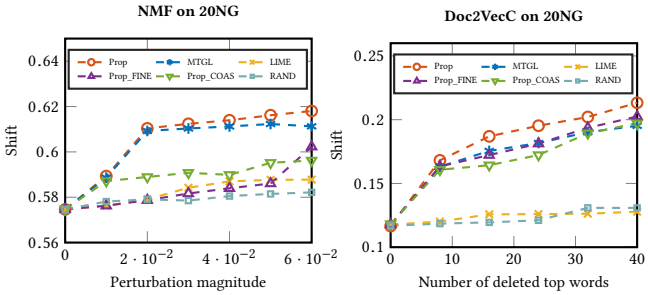

Figure 4: Evaluation of interpretation for document representation instances on 20NewsGroup data.

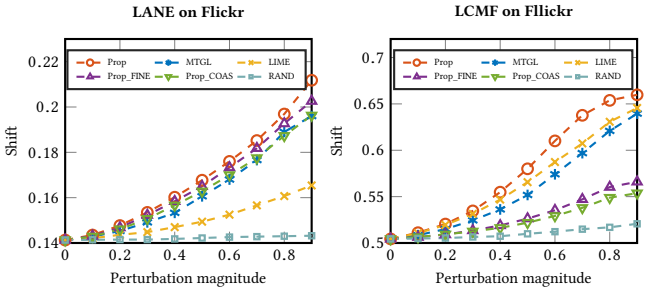

Figure 5: Evaluation of interpretation for network embedding instances on Flickr data.

used in NMF, the adversarial sample for instance n is generated through perturbing the attributes:

$$\mathbf{X}^*(n,:) = \mathbf{X}(n,:) - \epsilon \mathbf{d}_n / \|\mathbf{d}_n\|_1, \tag{14}$$

where ϵ is the perturbation magnitude, $\mathbf{d}_n \in \mathbb{R}^M$ and

$$\mathbf{d}_n(m) = \begin{cases} thre - rank(\mathbf{y}'_n(m)) + 1, & \text{if } rank(\mathbf{y}'_n(m)) \leq thre \\ 0, & \text{otherwise} \end{cases}. \tag{15}$$

To understand the equations above, $thre$ is the number of attributes to be perturbed, $\mathbf{y}'_n$ denotes the interpretation for object n, $rank(\cdot)$ represents the rank of the entry after sorting all vector entries in descending order. We set $thre = 10$ in experiments. Therefore, an attribute assigned with higher importance score by interpretation is subject to greater perturbation. However, for representation learning models such as Doc2VecC that accept raw texts (i.e., discrete attributes) as input, the way of generating adversarial samples is slightly different. We delete from texts the top words that are assigned with high importance scores by interpretation methods. The number of words to be deleted is $thre$. The resultant text document, after top words being deleted, is treated as the adversarial sample.

After adversarial perturbation, we append the adversarial samples to the original dataset, obtain the new representation vectors, and measure the relative *shift* of new vectors compared with their original vectors before perturbation. The shift of a representation vector is defined as below:

$$shift(\mathbf{z}, \epsilon) = 1 - \frac{|\text{Neighbors}(\mathbf{z}) \cap \text{Neighbors}(\mathbf{z}^*)|}{|\text{Neighbors}(\mathbf{z})|}, \tag{16}$$

where Neighbors($\mathbf{z}$) is the set of neighbors near vector $\mathbf{z}$ in the latent space. $\mathbf{z}^*$ denotes the new vector from representation learning after adversarial perturbation. We set the number of neighbors as $0.03 \times N$ in our experiments. It is expected that more accurate interpretation, when applied in adversarial attacks, will lead to greater shift of representation vectors.

The effectiveness of different interpretation methods in generating adversarial samples is shown in Figure 4. Some observations are given as below:

- The shift of representation instances is greater as we increase the perturbation degree, especially compared with random perturbation. This means adversarial perturbation, to some extent, can be used for evaluating whether interpretation unveils the patterns learned by representation learning. Here we include the random method to exclude the possibility that the shifts are caused by other unknown factors.
- The global interpretation methods, including the proposed method and MTGL, initiate more effective adversarial attacks than the local method LIME. It thus shows the advantage of understanding the representation results from a broader point of view. Meanwhile, the proposed method performs better than MTGL.
- The proposed method achieves better performance than its two variants. It means that no matter we cluster representation instances with coarse or fine granularity, combining them together can promote the interpretation accuracy. It justifies our choice of including multiple hierarchies in the spatial encoder.

5.2.2 Interpretation Evaluation for Attributed Network Embedding. We now evaluate the accuracy of interpretation for network embedding. Since the attribute values in this dataset are continuous, we adopt the same adversarial perturbation scheme as on TF-IDF attributes in the last experiment. The evaluation results are shown in Figure 5. Similar observation can be drawn as the last experiment, so we do not repeat them here. Furthermore, by comparing the plots in Figure 4 and Figure 5, we can find that in NMF-related models (i.e., LCMF on Flickr data and NMF on 20NG data), the performance of LIME is better than that in other scenarios. A possible reason is that NMF models, especially with L_1 regularization, distinguish different latent space regions by assigning large $\mathbf{z}(d)$ values of certain dimensions d to the representation instances within each region. Therefore, each group is more linearly separable to other groups.

5.3 Interpretation Evaluation Through Content-Based Recommendation

In this subsection, we evaluate the accuracy of interpreting latent representation of movie viewers obtained from collaborative filtering (CF). The evaluation strategy is different from the one used in the last subsection. The reason is that the attribute matrix $\mathbf{Y}$ is disjoint from the input information to CF, so we cannot apply adversarial attacks on $\mathbf{Y}$ to change the representation vectors.

In this experiment, we apply the idea of content-based recommendation to evaluate the interpretation for collaborative filtering. Specifically, after obtaining interpretation $\mathbf{y}'_n$ for the representation vector $\mathbf{Z}(n,:)$ of a user n, we use $\mathbf{y}'_n$ to recommend the movies that the user may be interested in. In another word, we use interpretation result as the input for content-based recommendation, since

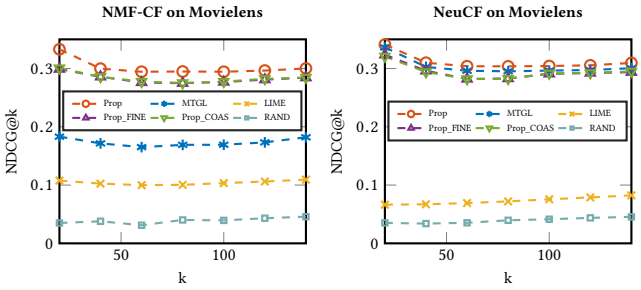

Figure 6: Interpretation performance evaluation through content-based recommendation.

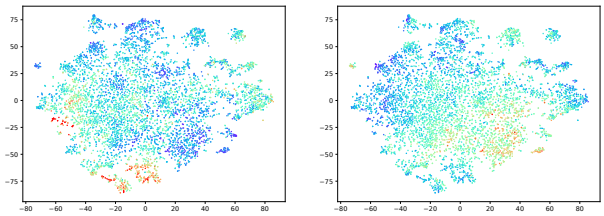

Figure 7: Visualization of attribute intensity over representation space. Left: The intensity of attribute "comedy"; Right: The intensity of attribute "tragedy". Red color means higher intensity, while blue color means lower intensity.

$\mathbf{y}'_n$ essentially acts as the profile of user n. The recommendation performance is measured as the similarity between the original CF result as ground-truth and the content-based recommendation result. We use nDCG@k as the metric. It is worth noting that the original testing ratings, if we have, for evaluating CF cannot be used to evaluate interpretation, because our goal is to measure whether interpretation is faithful with respect to the target CF model but not the testing ratings.

The motivation of using content-based recommendation as evaluation strategy is shown in Figure 7. The 2D space visualization, for representation vectors of movie viewers, is generated using t-SNE [25]. The color indicates the intensity of certain attribute (i.e., movie tag), where "red" means that the tag is intensively associated with the movies watched by the viewer, while "blue" means the opposite. From the figure, we observe that tag intensity is highly correlated with the distribution of representation vectors. This motivates us to use tag information as description to the interests of movie viewers located in different regions of the latent space. Besides, the highlighted area of tag "comedy" is disjoint from that of tag "tragedy", which makes sense as they are opposed in meaning.

We did experiments on two CF models, where one is based on NMF and the other is NeuCF built upon neural networks. The performance is shown in Figure 6. We can observe that the proposed framework is better than the baseline methods. LIME is not as advantageous as global interpretation methods, although the gap is smaller when interpreting NMF. The global interpretation methods have significantly better performance than the random method, which indicates that they could preserve the prediction mechanism of the original recommendation systems. The proposed framework, as well as its variants, have stable performances in different scenarios.

6 RELATED WORK

Representation learning maps data instances into an expressive and usually low-dimensional latent space, where similar instances are mapped close to each other [1]. Representation learning is fundamental to the success of downstream machine learning models, especially when dealing with data of raw formats such as images [41], texts [2, 26] and networks [29, 35]. Some models such as autoencoders [37] or embedding models [26, 29] explicitly aim at learning the representation of input data, while many applications such as multimodal learning [14], recommender systems [19] and anomaly detection [3] rely on effective representations for improving performance although they may not pursue learning representations as the direct goal. The diversity of the application scenarios and model structures put forward request on the flexibility of the corresponding interpretation framework.

Despite the importance of representation learning, similar to many complex prediction models, its interpretability has not received much attention in applications. Some preliminary methods have been developed to interpret machine learning models [8, 27]. Existing interpretation methods can be roughly divided into several categories. First, a major category of methods provide local interpretation for supervised models. Some methods build local explanable models to approach the behavior of the original model [30, 45], while some methods utilize interpretable components such as attention layers [32]. Some approaches use gradient-based measures to unveil the behavior of models as input changes [31]. Also, some model components can also be utilized to obtain faithful and reasonable interpretation, such as the middle layers in CNN [9], and piece-wise linear activation functions [5]. Second, some methods provide global interpretation and extract knowledge from supervised models. Global interpretation can be extracted in the form of rule-based decision trees [21]. In image classification, the knowledge could be expressed as a network connecting different visual concepts arranged according to their topological relations [44]. Third, some recent approaches try to extract global interpretation for unsupervised models. Some attempts include extracting a taxonomy to understand how overall network embedding instances distribute in the latent space [23], where cluster-level interpretation of instances is achieved. In addition, Sharma et al. initiate a study to analyze the geometric arrangement of embeddings in knowledge graphs [33]. Finally, some effort has been paid to improve the interface between ML systems and users, such as vector-space visualization algorithms like tSNE [25], and exploration about the explanation components that are most compelling to users in recommender systems [12].

7 CONCLUSION AND FUTURE WORK

In this paper, we design a new approach to interpret the outcome of representation learning. The problem involves two aspects. The first aspect is understanding how representation instances distribute in the latent space. The second aspect is to extract the semantic meaning of different regions in the distribution. To tackle the first issue, given a representation instance, a hierarchical spatial encoder is designed to express where it locates in the latent space. Following that, a multimodal autoencoder is built to learn the correlation between spatial codes and attributes. The recovered attributes, inferred from

the autoencoder given the spatial code of an representation instance, describe the noteworthy characteristics of the instance in the latent space.

The problem of representation interpretation has not been well-studied. Some possible future work is as follows. First, more complicated scenarios can be considered, where there exist multiple relations between instances such as in knowledge graphs. Second, we can explore how to combine representation interpretation with existing supervised machine learning interpretation methods. Third, techniques can be developed to feed interpretation back to the learning process to improve the model.

ACKNOWLEDGMENTS

The work is, in part, supported by DARPA (#N66001-17-2-4031) and NSF (#IIS-1657196, #IIS-1718840). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* (2013).
[2] Minmin Chen. 2017. Efficient vector representation for documents through corruption. *arXiv preprint arXiv:1707.02377* (2017).
[3] Ting Chen, Lu-An Tang, Yizhou Sun, Zhengzhang Chen, and Kai Zhang. 2016. Entity embedding-based anomaly detection for heterogeneous categorical events. *arXiv preprint arXiv:1608.07502* (2016).
[4] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *NIPS*.
[5] Lingyang Chu, Xia Hu, Juhua Hu, Lanjun Wang, and Jian Pei. 2018. Exact and Consistent Interpretation for Piecewise Linear Neural Networks: A Closed Form Solution. *arXiv preprint arXiv:1802.06259* (2018).
[6] Chris Ding, Xiaofeng He, and Horst D Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*. SIAM.
[7] Chris Ding, Tao Li, Wei Peng, and Haesun Park. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*.
[8] Mengnan Du, Ninghao Liu, and Xia Hu. 2018. Techniques for Interpretable Machine Learning. *arXiv preprint arXiv:1808.00033* (2018).
[9] Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. 2018. Towards Explanation of DNN-based Prediction with Guided Feature Inversion. In *KDD*.
[10] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* (2016).
[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[12] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*.
[13] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *WSDM*. ACM.
[14] Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.
[15] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. 2016. Examples are not enough, learn to criticize! criticism for interpretability. In *NIPS*.
[16] Seyoung Kim and Eric P Xing. 2010. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*.
[17] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[18] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning*.
[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[20] Da Kuang, Sangwoon Yun, and Haesun Park. 2015. SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering. *Journal of Global Optimization* (2015).
[21] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *KDD*.
[22] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*. 556–562.
[23] Ninghao Liu, Xiao Huang, Jundong Li, and Xia Hu. 2018. On Interpretation of Network Embedding via Taxonomy Induction. In *KDD*.
[24] Ninghao Liu, Hongxia Yang, and Xia Hu. 2018. Adversarial Detection with Model Interpretation. In *KDD*.
[25] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* (2008).
[26] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[27] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2017. Methods for Interpreting and Understanding Deep Neural Networks. *arXiv preprint arXiv:1706.07979* (2017).
[28] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *ICML*.
[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM.
[30] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *KDD*.
[31] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.. In *ICCV*.
[32] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*.
[33] Aditya Sharma, Partha Talukdar, et al. 2018. Towards Understanding the Geometry of Knowledge Graph Embeddings. In *ACL*.
[34] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. 2016. Visualizing large-scale and high-dimensional data. In *WWW*.
[35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*.
[36] Robert Tibshirani and Guenther Walther. 2005. Cluster validation by prediction strength. *Journal of Computational and Graphical Statistics* (2005).
[37] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*.
[38] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* (2010).
[39] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* (2007).
[40] Fei Wang, Tao Li, Xin Wang, Shenghuo Zhu, and Chris Ding. 2011. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery* (2011).
[41] Faqiang Wang, Wangmeng Zuo, Liang Lin, David Zhang, and Lei Zhang. 2016. Joint learning of single-image and cross-image representations for person re-identification. In *CVPR*.
[42] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*.
[43] Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. 2015. Exploring Implicit Hierarchical Structures for Recommender Systems.. In *IJCAI*.
[44] Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. 2017. Interpreting cnn knowledge via an explanatory graph. *arXiv preprint arXiv:1708.01785* (2017).
[45] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. 2018. Interpretable convolutional neural networks. In *CVPR*.
[46] Zhenyue Zhang and Hongyuan Zha. 2004. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing* (2004).
[47] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative location and activity recommendations with GPS history data. In *WWW*.