TRAP: Two-level Regularized Autoencoder-based Embedding for Power-law Distributed Data

Dongmin Park, Hwanjun Song, Minseok Kim, Jae-Gil Lee∗
Graduate School of Knowledge Service Engineering, KAIST
{dongminpark,songhwanjun,minseokkim,jaegil}@kaist.ac.kr

ABSTRACT
Recently, autoencoder (AE)-based embedding approaches have achieved state-of-the-art performance in many tasks, especially in top-k recommendation with user embedding or node classification with node embedding. However, we find that many real-world data follow the power-law distribution with respect to the data object sparsity. When learning AE-based embeddings of these data, dense inputs move away from sparse inputs in an embedding space even when they are highly correlated. This phenomenon, which we call polarization, obviously distorts the embedding. In this paper, we propose TRAP that leverages two-level regularizers to effectively alleviate the polarization problem. The macroscopic regularizer generally prevents dense input objects from being distant from other sparse input objects, and the microscopic regularizer individually attracts each object to correlated neighbor objects rather than uncorrelated ones. Importantly, TRAP is a meta-algorithm that can be easily coupled with existing AE-based embedding methods with a simple modification. In extensive experiments on two representative embedding tasks using six real-world datasets, TRAP boosted the performance of the state-of-the-art algorithms by up to 31.53% and 94.99% respectively.

CCS CONCEPTS
• Computing methodologies → Regularization; Neural networks; • Information systems → Data mining.

KEYWORDS
Power-law Distribution, Recommender System, Graph Embedding, Autoencoder

ACM Reference Format:

1 INTRODUCTION
With the rapid growth of online services including e-commerce and social media, a variety of high-dimensional datasets have become available, e.g., user-item transaction matrices and social relation matrices [17]. However, owing to their extremely high dimensionality, most of existing machine learning algorithms have been reported to suffer from high computational and space complexity [1]. Thus, many researchers have employed a data embedding technique, which maps high-dimensional data to lower-dimensional data, in order to directly apply the existing algorithms on the lower-dimensional data [8, 15, 29].

Recently, numerous autoencoder (AE)-based embedding approaches have been studied actively, and they are empirically proven to achieve not only low-dimensional but also highly informative embeddings because of the strong representation power of neural networks. Hence, it becomes feasible to capture a data manifold smoothly even in the high-dimensional data [4]. This family of AE-based approaches is well known to reach the state-of-the-art performance in numerous machine learning tasks, especially in top-k recommendation tasks with user embedding [13, 29, 30] or link prediction and node classification (or clustering) tasks with node embedding [4, 5, 8].

Despite their great success, we claim that a skewed sparsity distribution of input vectors (e.g., movie ratings) in real-world data severely hurts the performance of embedding methods. Specifically, as shown in Figure 1a, the sparsity distribution of input vectors is strongly skewed toward being sparse. This observation is natural in considering that various phenomena in the real world approximately follow a power law over a wide range of magnitudes [3]. However, as shown in Figure 1b, this skewed distribution entails a polarization problem that causes a dense input to move away from sparse inputs in an embedding space (See Section 3.2 for the details), even when the two inputs are highly correlated. Evidently, these polarized latent representations degrade the performance of the embedding methods.

Intuitively, the deleterious effect of polarization is very common in real-world scenarios. Let’s consider a recommendation task where its data follows the power-law distribution [28]. The goal of the task is to suggest unexperienced but interesting items to users based on their estimated preference under the assumption that people’s tastes are highly correlated with each other. However, the polarization problem hinders dense inputs from being closely located to sparse inputs in the latent space regardless of the correlation between them, and thus, the existing embedding methods may inaccurately estimate the users’ preference. This limitation calls for a new approach to alleviate the polarization problem.

To handle the polarization problem, we propose a novel meta-algorithm called TRAP (Two-level Regularized Autoencoder based embedding for Power-law distributed data) that can be easily combined with any existing autoencoder-based embedding methods leveraging two-level regularizers. (i) The “macroscopic regularizer” generally prevents dense input objects from being distant from...
2 RELATED WORK

Numerous studies have been conducted to learn the low-dimensional embeddings from sparse data. Here, we briefly review the studies for two representative tasks: (i) user embedding in recommender systems and (ii) node embedding in graph mining. Note that most of the existing work has overlooked the polarization problem resulting from the inherent nature that the density of real-world inputs follows the power-law distribution. As far as we know, TRAP is the first method to overcome the polarization problem in the embedding task.

2.1 User-Item Embedding

Learning a low-dimensional embedding of users and items has been reported to make a considerable performance improvement in the top-k recommendation task [13, 30]. A popular approach is matrix factorization (MF) [12] that decomposes the sparse user-item rating matrix into the product of two lower-dimensional but dense matrices based on the singular value decomposition (SVD). To further improve the typical MF, Koren [11] integrated the neighbor-based method, and Mnih and Salakhutdinov [16] employed the probabilistic regularization technique. Nevertheless, their performances are limited by the linearity of the SVD [13].

By virtue of the non-linearity in neural networks, many recent researchers have attempted to design AE-based embedding approaches [29]. AutoRec [21] reconstructs only the observed ratings in a sparse user-item rating matrix based on the point-wise loss. CDAE [27] adopts the denoising AE [25] to force the hidden layer to discover more robust embeddings from the sparse input. Multi-VAE [13] adopts the variational AE (VAE) [10] and uses a multinomial log-likelihood loss, which is known to suit the top-k recommendation. JCA [30] introduces a joint learning paradigm with their pair-wise loss such that the AE model captures the correlation between users and items.

2.2 Node Embedding

Mapping nodes in a graph to the low-dimensional embeddings is another essential problem in numerous tasks such as link prediction and node classification. Many research efforts have been devoted to encoding the nodes in the embedding space. DeepWalk [19] feeds the truncated random walks of the nodes into the SkipGram model [15] to preserve the high-order proximity between the nodes. As opposed to DeepWalk, LINE [24] focuses on preserving the first (and second) order proximity. Node2Vec [7] introduces both breadth-first and depth-first node sampling methods to construct the context of nodes. Struc2Vec [20] generates a series of the weighted auxiliary graphs and then uses the biased random walks as an input of Node2Vec. However, their general principle based on SkipGram fails to capture the high non-linearity in the graph [5].
Similar to the user-item embedding, recent studies [2, 4, 5, 26] have proposed AE-based embedding approaches to capture the non-linearity in the graph. SDNE [26] exploits both local and global graph structures to generate the node embeddings based on the first-order and second-order proximity. DANE [4] maintains an additional AE to reconstruct the sparse attribute of each node. ProGAN [5] exploits more complex underlying proximities between the nodes generated by a generative adversarial network (GAN) [6].

3 POLARIZATION PROBLEM

3.1 Preliminary

Let \( X = \{X_1, X_2, \ldots, X_M\} \in \mathbb{R}^{M \times N} \) be a matrix of \( M \) objects (i.e., row vectors in \( X \)), where the \( i \)-th object is \( X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,N}) \) with \( N \) properties (i.e., values in a vector \( X_i \)). Then, a low-dimensional embedding is formally defined by Definition 3.1.

Definition 3.1: A low-dimensional embedding is learning a function \( f: X \rightarrow X' \), where \( X, X' \in \mathbb{R}^{M \times N} \) is an input matrix, \( X' \in \mathbb{R}^{M \times N'} \) is an embedding matrix, and \( N > N' \).

For the notation of sparse embedding, the sparsity of the \( i \)-th object \( X_i \) is defined by Definition 3.2, where \( [\cdot] \) is the Iverson bracket.

Definition 3.2: The sparsity of the \( i \)-th object \( X_i \) is formulated as in Eq. (1).

\[
S(X_i) = \frac{\sum_{X_{i,j} \in X} [X_{i,j} = 0]}{\dim(X_i)} \quad \square
\]

Then, we regard that the matrix \( X \) is a sparse binary matrix by Definition 3.3, which is a common problem setting in the recent literature for both recommender systems and graph embedding [4, 13, 21, 26, 27, 30].

Definition 3.3: A matrix \( X \in \mathbb{R}^{M \times N} \) is a sparse binary matrix if Eq. (2) holds.

\[
(\forall X_{i,j} \in X : X_{i,j} \in \{0, 1\}) \wedge (1/M) \sum_{i=1}^{M} S(X_i) = 1 \quad \square
\]

3.2 Theoretical Analysis on Polarization

Let \( X = (x_1, x_2, \ldots, x_N) \) be a random vector where each element \( x_j \in X \) is a binary random variable such that \( P(x_j = 1) = p_x \) and \( P(x_j = 0) = (1 - p_x) \). Besides, let \( f(X) \) and \( f(X) \) be the embedding vectors of two random vectors \( X_s \) and \( X_d \) with different sparsity such that \( E[f(X_s)] > E[f(X_d)] \). Then, the notion of polarization is formalized as Definition 3.4, in which the total variance \[18, 23\] in Definition 3.5 is used as a measure of the dispersion from a center \( E[f(X)] \) to a given embedding vector \( f(X) \). That is, the higher the total variance, the more \( f(X) \) is dispersed from the center.

Definition 3.4: Polarization is a phenomenon that the embedding vector of dense inputs tends to be located farther from the center than that of sparse inputs, which is proven by Theorem 3.9.

Definition 3.5: The total variance of an embedding vector \( f(X) \) is the sum of all eigenvalues \( \lambda \) of the cov(\( f(X) \)) as in Eq. (3).

\[
\text{totvar}(f(X)) = \sum_{i=1}^{N'} \lambda_i = \text{tr}(\text{cov}(f(X))) \quad \square
\]

We theoretically prove the existence of polarization, assuming more realistic network architectures step by step: a 1-layer linear AE (Lemma 3.6), a 1-layer non-linear AE (Lemma 3.7), and a multi-layer non-linear AE (Lemma 3.8).

Lemma 3.6: Let \( f(X) = WX + b \) be a 1-layer AE with \( N' \) hidden units and an identity activation function, where \( W \in \mathbb{R}^{N \times N} \) and \( b \in \mathbb{R}^{N} \). Then, Eq. (4) holds.

\[
E[S(X_i)] > E[S(X_d)] \rightarrow \text{totvar}(f(X_s)) < \text{totvar}(f(X_d)) \quad (4)
\]

Proof. By Definition 3.5, the total variance of the embedding vector \( f(X) \) is derived by Eq. (5).

\[
\text{totvar}(f(X)) = \text{tr}(\text{cov}(WX + b)) = \text{tr}(\text{cov}(WX)) = \text{tr}(W \text{cov}(X)W^T) = \text{tr}(\text{cov}(X)W^TW) \quad (5)
\]

Since each element \( x_j \in X \) is a binary random variable, \( x_j \) follows a Bernoulli distribution \( B(1, p_x) \), where \( p_X = (1 - E[S(X)]) \). Then, by \( E[S(X_s)] > E[S(X_d)] > 0 < p_{X_s} < p_{X_d} < 1/2 \), and both \( \text{cov}(X_s) - \text{cov}(X_d) \) and \( W^TW \) are positive semidefinite. Therefore, Eq. (6) holds because \( \text{tr}(AB) > 0 \) if \( A \) and \( B \) are positive semidefinite. This concludes the proof.

\[
\text{totvar}(f(X_s)) - \text{totvar}(f(X_d)) = \text{tr}(\text{cov}(X_s)W^TW - \text{cov}(X_d)W^TW) > 0 \quad \square
\]

Lemma 3.7: Let \( \text{ReLU}(Z) = \max(0, Z) \) be a rectified linear unit (ReLU) activation function, where \( Z = f(X) \) is a 1-layer embedding. Then, Eq. (7) holds.

\[
\text{totvar}(Z_s) < \text{totvar}(Z_d) \rightarrow \text{totvar}(\text{ReLU}(Z_s)) < \text{totvar}(\text{ReLU}(Z_d)) \quad (7)
\]

Proof. Let’s assume that \( Z_s \) and \( Z_d \) follow the multivariate normal distribution, where \( E[Z_s] = E[Z_d] \) and \( \text{cov}_{i,j}(Z_s) < \text{cov}_{i,j}(Z_d) \) for all \( i \)-th diagonal elements of the covariance matrices.

1. \( E[Z_s] \geq 0 \): Let \( Y = \min(Z_s, 0) \), and \( U = Z \) if \( Z_s \geq 2E[Z_s] \), \( E[Z_s] \) if \( 2E[Z_s] > Z \geq 0 \), and \( Y \) otherwise. Then, \( \text{cov}_{i,j}(\text{ReLU}(Z)) = \text{cov}_{i,j}(Z) - \text{cov}_{i,j}(Y) \). Since \( E[U] = E[Z_s] = E[Z_d] \), by the definition of the covariance, Eq. (8) holds.

\[
\text{cov}_{i,j}(\text{ReLU}(Z_s)) - \text{cov}_{i,j}(\text{ReLU}(Z_d)) = \text{cov}_{i,j}(Z_s) - \text{cov}_{i,j}(Z_d) - \text{cov}_{i,j}(Y_s) - \text{cov}_{i,j}(Y_d) \quad (8)
\]

(ii) \( E[Z_s] < 0 \): The proof is similar to the above case.

Therefore, \( \text{cov}_{i,j}(\text{ReLU}(Z_s)) > \text{cov}_{i,j}(\text{ReLU}(Z_d)) \) for every \( i \)-th diagonal element. This concludes the proof.

Lemma 3.8: Let’s consider \( f'(Z') = W'Z' + b' \), where \( W' \in \mathbb{R}^{N' \times N''} \), \( b' \in \mathbb{R}^{N''} \), and \( Z' \) is the embedding passing through \( \text{ReLU}(\cdot) \) in the previous layer. Then, Eq. (9) holds.

\[
\forall_{i} \text{cov}_{i,j}(Z'_s) < \text{cov}_{i,j}(Z'_d) \quad \rightarrow \quad \text{totvar}(f'(Z'_s)) < \text{totvar}(f'(Z'_d)) \quad (9)
\]

Proof. By Eq. (6) and the condition of Eq. (9), Eq. (10) holds.

\[
\text{totvar}(f'(Z'_s)) - \text{totvar}(f'(Z'_d)) = \text{tr}((\text{cov}(Z'_s) - \text{cov}(Z'_d))W'^TW'') > 0 \quad \square
\]
Finally, the overall proof of the polarization on a multi-layer non-linear AE is concluded by Theorem 3.9.

**Theorem 3.9.** Let $f(X)$ be a multi-layer AE with the ReLU activation function. Then, $\text{totvar}(f(X_d)) > \text{totvar}(f(X_s))$.

Proof. By the mathematical induction in which Lemma 3.6 is the base step and Lemmas 3.7 and 3.8 are the consecutive inductive steps, the polarization still holds. 

Figure 2 shows the distance from the center to the embedding vector according to the number of a user’s ratings in the MovieLens-1M dataset when using AutoRec with two layers. Similar to the result of AutoRec with one layer in Figure 1b, the users with dense ratings tended to be located farther than those with sparse ratings. That is, we empirically confirm Theorem 3.9.

4 TWO REGULARIZERS OF TRAP

The key idea of TRAP is to constrain the distance between sparse and dense objects so that they are not too far from each other, which is achieved by two simple and effective regularizers: (i) the macroscopic regularizer that generally restricts dense objects being away from sparse objects, and (ii) the microscopic regularizer that individually adjusts each object’s movement to neighbor with correlated objects.

4.1 Macroscopic Regularizer

To handle the polarization problem, an intuitive method is to constrain the dispersion of data objects by pulling them into the center of their correlated group in the embedding space. However, this process is not straightforward because the clustering task may induce several difficulties including high computation cost and low quality of the result [22].

In this regard, we introduce a macroscopic regularizer that simply pulls all objects into the origin (i.e., zero vector) in the embedding space. Specifically, as shown in Eq. (11), this regularizer adds the total $L_2$-norm of all embeddings $z$ as a penalty term with its scaling hyperparameter $\eta$ in the objective function, where AE’ is any existing AE-based method, $L'$ is its reconstruction loss function, and $L$ is the number of layers of the encoder in AE’.

$$L = \sum_{i=1}^{M} (L'(X_i, AE'(X_i)) + \eta \sum_{l=1}^{L} \|z^{[l]}_i\|_2^2) \quad (11)$$

Since the dense objects which tend to be far from the origin have a higher $L_2$-norm than the sparse objects which tend to be close from the origin, the dense ones are pulled strongly whereas the sparse ones are pulled weakly. Consequently, the macroscopic regularizer generally restricts the dense objects being distant from other sparse objects.

4.2 Microscopic Regularizer

One limitation of the macroscopic regularizer lies in that it is applied invariably to the input data as parameter update is shared by all objects, while each object is not equally distant from the rest correlated objects in the embedding space. To achieve the goal of embedding which is to locate correlated objects nearby, we additionally introduce a microscopic regularizer that adjusts each object’s sparsity by adding non-shared parameters $\nu$ called object-wise scaling parameters, as shown in Figure 3. More specifically, as shown in Eq. (12), the object-wise scaling parameter $\nu_l^i$ of the object $X_i$ is multiplied with the output of neurons in the $l$-th layer before the non-linear activation $\sigma$, where $\sigma$ is the element-wise product.

$$z^{[l]}_{i} = \begin{cases} \sigma((W^{[l]}X_i + b^{[l]}) \circ \nu_l^{[l]}), & \text{if } l = 1 \\ \sigma((W^{[l]}z^{[l-1]}_{i} + b^{[l]}) \circ \nu_l^{[l]}), & \text{otherwise} \end{cases} \quad (12)$$

That is, when applied with the microscopic regularizer, the microscopic regularizer provides additional capacity to let an input object become closer to its correlated objects in an object-wise manner.

4.3 Quick Analysis on TRAP

We contend that the polarization problem in Definition 3.4 can be alleviated by TRAP based on the following explanation. Let’s recall Lemma 3.6, but the AE function is now defined as $f'(X) = (Wx + b) \circ \nu$ with the $\nu$ of the microscopic regularizer. Then, Eq. (6) is changed to Eq. (13).

$$\text{totvar}(f'(X_d)) - \text{totvar}(f'(X_s)) = tr(\text{cov}(X_d)(W \circ \nu_d)^T(W \circ \nu_d)) - tr(\text{cov}(X_s)(W \circ \nu_s)^T(W \circ \nu_s)) \quad (13)$$

In this case, note that Lemma 3.6 does not always hold because the total variance of the embedding vector $f'(X_d)$ can be rather smaller than that of the embedding vector $f'(X_s)$ if $\nu_d$ of the dense object $X_d$ is much smaller than $\nu_s$ of the sparse object $X_s$. With the

---

**Figure 2:** Polarization problem when using AutoRec with two layers on the MovieLens-1M dataset.

**Figure 3:** Structure of TRAP. The red (or blue) neuron means a “non-shared” object-wise scaling parameter which is assigned to each object.

---

2 We used the tanh function as the activation in all experiments.
macroscopic regularizer, \(\nu_M\) was generally much smaller than \(\nu_S\) according to our ablation study in Section 5.1.6. Therefore, TRAP can indeed overcome the polarization problem.

## 5 EXPERIMENTS

To validate the superiority of TRAP, we performed extensive experiments on two independent tasks: (i) user-item embedding and (ii) node embedding. TRAP as well as other algorithms were implemented using TensorFlow 1.8.0 and executed using a single NVIDIA Titan Volta GPU. For reproducibility, we provide the source code at https://github.com/kaist-dmlab/TRAP.

### 5.1 User-Item Embedding

#### 5.1.1 Datasets

We performed a user-item embedding task on three user feedback datasets: MovieLens-1M (ML1M)\(^3\), Yelp\(^4\), and VideoGame (Video)\(^5\). Here, the rating or purchase history of the user \(i\) for the item \(j\) corresponds to the value of \(x_{i,j} \in \mathbb{R}\). For the embedding task, we converted all users’ ratings or histories into binary values \(\in \{0, 1\}\) following the problem setting in Section 3.1, where 0 is negative feedback and 1 is positive feedback. Specifically, in ML1M and Yelp, a user-item rating was converted to 1 if it is greater than or equal to 4 and to 0 otherwise. In VideoGame, all items purchased less than 5 times and all users who purchased less than 5 items were excluded by the pre-processing step, and then a user-item history was binarized to 1 if the user purchased the item and to 0 otherwise. In addition, for each dataset, we randomly selected 70% of all user-item values as the training set, and 10% of them as the validation set, and the rest 20% of them as the test set. Table 1 summarizes the statistics of each dataset. Note that our sparsity assumption in Definition 3.3 is likely to hold in real-world user feedback data as the sparsity of data is almost 1 in Table 1. Also, the positive skewness\(^6\) in Table 1 implies that the sparsity of each object approximately follows the power law.

#### 5.1.2 Algorithms

- **MF** [12]: A traditional matrix factorization model with mean square error (MSE) loss and alternating least squares (ALS) for the optimization.
- **NCF** [9]: A combination model of an extended neural network-based MF model and a multi-layer perceptron (MLP) model.
- **AutoRec** [21]: A 1-layer AE model using user rating vectors as inputs. Because the loss of AutoRec is devised for the explicit feedback, we converted it into the pairwise loss in JCA, which is proven to be more effective in top-k recommendation with implicit feedback.
- **CDAE** [27]: A 1-layer denoising AE model. We chose the hinge-based pairwise loss introduced in CDAE.
- **MultiVAE** [13]: A multi-layer VAE model. We used the multinomial probabilistic loss presented in MultiVAE.
- **JCA** [30]: A joint model of user-based AutoRec and item-based AutoRec that fully takes advantage of user-item correlation. We used the pairwise loss proposed in JCA.

We combined TRAP with AutoRec, CDAE, MultiVAE, and JCA, each of which is denoted as TRAP\(_{Auto}\), TRAP\(_{CDAE}\), TRAP\(_{Multi}\), and TRAP\(_{JCA}\), respectively. We validated the performance improvement of the combined ones compared with the original ones.

#### 5.1.3 Experiment Setting

The hyperparameters of all compared algorithms were favorably set to be the best values reported in the original papers [9, 12, 13, 21, 27, 30]. Regarding TRAP, the weight \(\eta\) for the macroscopic regularizer was set to be the best value found by a grid \(\eta \in \{3 \times 10^{-3}, 1 \times 10^{-3}, 3 \times 10^{-5}, 1 \times 10^{-5}, 1 \times 10^{-2}\}\), and the object-wise weight \(\nu\) for the microscopic regularizer was initialized as the values randomly drawn from a uniform distribution \(U(0, 1)\). As for the training configuration, we used an Adam optimizer, a mini-batch size of 1500, and a constant learning rate of 0.003.

#### 5.1.4 Evaluation Metrics

To measure the performance of the user-item embedding, we performed a top-\(k\) recommendation task, which is a popular way to validate the quality of obtained user embeddings [29]. The top-\(k\) recommendation is to suggest \(k\) unseen items to each user. Let \(I_i^*\) and \(I_i(k)\) be the ground truth and \(k\) recommended items for the \(i\)-th user, where \(|I_i^*| \geq k\). Then, given \(M\) users and \(N\) items, the recommendation performance is typically measured by the following four metrics:

\[
\text{Precision}_{@k} = \frac{1}{M} \sum_{i=1}^{M} \frac{|I_i^* \cap \hat{I}_i(k)|}{k};
\]

\[
\text{Recall}_{@k} = \frac{1}{M} \sum_{i=1}^{M} \frac{|I_i^* \cap \hat{I}_i(k)|}{|I_i^*|};
\]

\[
\text{F1-score}_{@k} = \frac{(2 \cdot \text{Precision}_{@k} \cdot \text{Recall}_{@k})}{(\text{Precision}_{@k} + \text{Recall}_{@k})};
\]

\[
\text{NDCG}_{@k} = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{\log_2 \left(\sum_{j \in \hat{I}_i(k)} \log_2(\text{rank}(j^*)) + 2\right)};
\]

where \(\text{NDCG}(\hat{I}_i^*) = \sum_{l=1}^{k} \frac{1}{\log_2(l)}\).

In support of reliable evaluation, we repeated every task five times and reported the average of each metric.

#### 5.1.5 Performance Comparison

Table 2 shows \(\text{F1-score}_{@k}\) and \(\text{NDCG}_{@k}\) on three datasets with varying \(k\). Overall, the performance of all existing AE-based embedding methods was significantly improved by incorporating TRAP into them regardless of \(k\) values. Quantitatively, compared with the original method, \(\text{F1-score}_{@k}\) was improved by up to 20.38% in ML1M, 24.16% in Yelp, and 31.53% in VideoGame; \(\text{NDCG}_{@k}\) was improved by up to 21.42% in ML1M, 21.51% in Yelp, and 25.76% in VideoGame. In particular, the best performance improvement was mostly achieved with CDAE. However, when combined with MultiVAE, the improvement

---

\(^3\)http://files.grouplens.org/datasets/movielens/ml-1m.zip

\(^4\)http://www.yelp.com/dataset/challenge

\(^5\)http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/ratings_VideoGames.csv

\(^6\)The skewness was measured by Pearson’s moment coefficient of skewness.
was relatively low because the effect of our macroscopic regularizer overlaps with that of the Gaussian prior \(N(0, 1)\) constraint to the embeddings in the original MultiVAE model. Nevertheless, the synergistic effect with our macroscopic regularizer is still effective. Meanwhile, the best performance on the two metrics was achieved by either \(\text{TRAP}_{\text{CDAE}}\) or \(\text{TRAP}_{\text{JCA}}\) in all datasets. Figures 4 and 5 show \(\text{precision}@k\) and \(\text{recall}@k\) of all algorithms on three datasets with varying \(k\). The performance trends with them were similar to those with \(\text{F1-score}@k\) and \(\text{NDCG}@k\). Most importantly, this consistent dominance of the combined methods empirically proves that mitigating the polarization problem significantly improves the performance of the embedding task.

### 5.1.6 Ablation Study

To individually examine the effect of the macroscopic and microscopic regularizers, we conducted additional ablation experiments on two methods, \(\text{AutoRec}\) and \(\text{CDAE}\). We first show the performance of each method’s vanilla version and then enable the components of \(\text{TRAP}\) to solely evaluate the effectiveness of each component. We denote the versions combined with

![Figure 4: Precision@k of all user-item embedding algorithms on three datasets.](image1)

![Figure 5: Recall@k of all user-item embedding algorithms on three datasets.](image2)

Table 2: \(\text{F1-score}@k\) and \(\text{NDCG}@k\) of all user-item embedding algorithms on three datasets (the best results are marked in bold).

<table>
<thead>
<tr>
<th>Metrics</th>
<th>ML1M</th>
<th>Yelp</th>
<th>VideoGame</th>
<th>ML1M</th>
<th>Yelp</th>
<th>VideoGame</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F1-score</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models</td>
<td>@1</td>
<td>@5</td>
<td>@10</td>
<td>@1</td>
<td>@5</td>
<td>@10</td>
</tr>
<tr>
<td>MF</td>
<td>0.0469</td>
<td>0.1418</td>
<td>0.1801</td>
<td>0.1410</td>
<td>0.0318</td>
<td>0.0343</td>
</tr>
<tr>
<td>NCF</td>
<td>0.0511</td>
<td>0.1525</td>
<td>0.1887</td>
<td>0.1555</td>
<td>0.0337</td>
<td>0.0354</td>
</tr>
<tr>
<td>AutoRec</td>
<td>0.0869</td>
<td>0.1486</td>
<td>0.1956</td>
<td>0.1676</td>
<td>0.0334</td>
<td>0.0357</td>
</tr>
<tr>
<td>TRAP</td>
<td>0.0968</td>
<td>0.1644</td>
<td>0.2051</td>
<td>0.1885</td>
<td>0.0366</td>
<td>0.0396</td>
</tr>
<tr>
<td>%improve</td>
<td>13.83%</td>
<td>10.61%</td>
<td>4.85%</td>
<td>10.89%</td>
<td>9.55%</td>
<td>11.14%</td>
</tr>
<tr>
<td><strong>NDCG</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models</td>
<td>@1</td>
<td>@5</td>
<td>@10</td>
<td>@1</td>
<td>@5</td>
<td>@10</td>
</tr>
<tr>
<td>MF</td>
<td>0.2778</td>
<td>0.2510</td>
<td>0.2472</td>
<td>0.0351</td>
<td>0.0372</td>
<td>0.0477</td>
</tr>
<tr>
<td>NCF</td>
<td>0.2955</td>
<td>0.2727</td>
<td>0.2709</td>
<td>0.0378</td>
<td>0.0390</td>
<td>0.0496</td>
</tr>
<tr>
<td>AutoRec</td>
<td>0.3197</td>
<td>0.2813</td>
<td>0.2735</td>
<td>0.0394</td>
<td>0.0408</td>
<td>0.0493</td>
</tr>
<tr>
<td>TRAP</td>
<td>0.3600</td>
<td>0.3126</td>
<td>0.2950</td>
<td>0.0413</td>
<td>0.0449</td>
<td>0.0551</td>
</tr>
<tr>
<td>%improve</td>
<td>31.97%</td>
<td>21.97%</td>
<td>20.57%</td>
<td>21.13%</td>
<td>21.32%</td>
<td>20.36%</td>
</tr>
</tbody>
</table>

Table 3: Ablation study for the two regularizers of \(\text{TRAP}\) (the best results are marked in bold).

<table>
<thead>
<tr>
<th>Models</th>
<th>ML1M</th>
<th>Yelp</th>
<th>VideoGame</th>
</tr>
</thead>
<tbody>
<tr>
<td>MF</td>
<td>0.0366</td>
<td>0.0386</td>
<td>0.0488</td>
</tr>
<tr>
<td>NCF</td>
<td>0.0366</td>
<td>0.0396</td>
<td>0.0482</td>
</tr>
<tr>
<td>AutoRec</td>
<td>0.0445</td>
<td>0.0458</td>
<td>0.0561</td>
</tr>
<tr>
<td>TRAP</td>
<td>0.0420</td>
<td>0.0451</td>
<td>0.0543</td>
</tr>
<tr>
<td>%improve</td>
<td>18.20%</td>
<td>21.13%</td>
<td>15.81%</td>
</tr>
<tr>
<td>MF</td>
<td>0.0426</td>
<td>0.0452</td>
<td>0.0539</td>
</tr>
<tr>
<td>NCF</td>
<td>0.0426</td>
<td>0.0452</td>
<td>0.0539</td>
</tr>
<tr>
<td>AutoRec</td>
<td>0.0426</td>
<td>0.0452</td>
<td>0.0539</td>
</tr>
<tr>
<td>TRAP</td>
<td>0.0426</td>
<td>0.0452</td>
<td>0.0539</td>
</tr>
<tr>
<td>%improve</td>
<td>18.00%</td>
<td>21.00%</td>
<td>15.70%</td>
</tr>
<tr>
<td>MF</td>
<td>0.0426</td>
<td>0.0452</td>
<td>0.0539</td>
</tr>
<tr>
<td>NCF</td>
<td>0.0426</td>
<td>0.0452</td>
<td>0.0539</td>
</tr>
<tr>
<td>AutoRec</td>
<td>0.0426</td>
<td>0.0452</td>
<td>0.0539</td>
</tr>
<tr>
<td>TRAP</td>
<td>0.0426</td>
<td>0.0452</td>
<td>0.0539</td>
</tr>
<tr>
<td>%improve</td>
<td>17.70%</td>
<td>20.70%</td>
<td>15.50%</td>
</tr>
</tbody>
</table>
Table 4: Performance comparison of TRAP$^{macro}$ with two normalization methods (the best results are marked in bold).

<table>
<thead>
<tr>
<th>Models</th>
<th>ML1M @1</th>
<th>@5</th>
<th>@10</th>
<th>Yelp @1</th>
<th>@5</th>
<th>@10</th>
<th>VideoGame @1</th>
<th>@5</th>
<th>@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoRec</td>
<td>0.949</td>
<td>0.946</td>
<td>0.956</td>
<td>0.167</td>
<td>0.334</td>
<td>0.357</td>
<td>0.0192</td>
<td>0.572</td>
<td>0.248</td>
</tr>
<tr>
<td>Sphere-norm</td>
<td>0.945</td>
<td>0.911</td>
<td>0.941</td>
<td>0.0055</td>
<td>0.0101</td>
<td>0.111</td>
<td>0.0044</td>
<td>0.0055</td>
<td>0.0057</td>
</tr>
<tr>
<td>L1-norm</td>
<td>0.832</td>
<td>0.858</td>
<td>0.936</td>
<td>0.048</td>
<td>0.332</td>
<td>0.360</td>
<td>0.0190</td>
<td>0.260</td>
<td>0.242</td>
</tr>
<tr>
<td>TRAP$^{macro}$</td>
<td>0.956</td>
<td>0.958</td>
<td>0.909</td>
<td>0.0173</td>
<td>0.3145</td>
<td>0.378</td>
<td>0.0223</td>
<td>0.292</td>
<td>0.264</td>
</tr>
<tr>
<td>CDAE</td>
<td>0.957</td>
<td>0.957</td>
<td>0.873</td>
<td>0.0155</td>
<td>0.323</td>
<td>0.352</td>
<td>0.0235</td>
<td>0.291</td>
<td>0.259</td>
</tr>
<tr>
<td>Sphere-norm</td>
<td>0.943</td>
<td>0.992</td>
<td>1.134</td>
<td>0.0064</td>
<td>0.159</td>
<td>0.130</td>
<td>0.0093</td>
<td>0.193</td>
<td>0.219</td>
</tr>
<tr>
<td>L1-norm</td>
<td>0.545</td>
<td>0.530</td>
<td>0.918</td>
<td>0.0156</td>
<td>0.330</td>
<td>0.353</td>
<td>0.0222</td>
<td>0.300</td>
<td>0.264</td>
</tr>
<tr>
<td>TRAP$^{macro}$</td>
<td>CDAE</td>
<td>0.961</td>
<td>0.953</td>
<td>0.929</td>
<td>0.0160</td>
<td>0.333</td>
<td>0.354</td>
<td>0.0233</td>
<td>0.306</td>
</tr>
</tbody>
</table>

Figure 6: Distribution of the object-wise scaling parameter $v_i$ according to the number of movie ratings of each user $i$ when training TRAP$^{auto}$ on the ML1M dataset.

Both regularizers as TRAP$^{auto}$ and TRAP$^{CDAE}$, respectively. Combination with either the macroscopic or microscopic regularizer is additionally indicated by the superscript, either macro or micro, on the name of each combined method. Table 3 reports the F1-score@k evaluation results of TRAP$^{auto}$ and TRAP$^{CDAE}$. For all three datasets and $k$ values, each regularizer leads to significant performance improvement. Moreover, the best performance improvement is achieved when both regularizers are combined. This result indicates that our proposed two-level regularizers have the synergistic effect to alleviate the polarization problem.

5.1.7 Macroscopic Regularizer vs. Other Normalizations. To validate the advantage of our macroscopic regularizer (i.e., l2-normalization) over other normalization methods, we conducted additional experiments with l1-normalization and hypersphere-normalization which force all embeddings to be located on the surface of a unit hypersphere. Table 4 shows F1-score@k of three normalization methods combined with AutoRec and CDAE on three datasets. Overall, our macroscopic regularizer always showed the highest improvement for both AutoRec and CDAE. The l1-normalization improved the performance, but its effect was weaker than that of our macroscopic regularizer; the hypersphere-normalization was shown to even degrade the performance.

5.1.8 Effect of Microscopic Regularizer. Furthermore, we plot the distribution of the object-wise scaling parameter $v_i$ with respect to the object’s input sparsity (i.e., the number of movie ratings) on ML1M, as shown in Figure 6. As an input object became denser, the parameter value decreased. By virtue of our design principle in Section 4.2, a large $v_i$ of the sparse objects forces them to move away from the origin, while a small $v_i$ of the dense objects forces them to stay close to the origin. Therefore, $v_i$ properly makes the correlated objects close in the embedding space, regardless of their input sparsity. Figure 7 represents the TSNE visualizations [14] over the learned embeddings of AutoRec and TRAP$^{auto}$. The change from Figure 7a to Figure 7b shows that polarization was effectively resolved by combining with TRAP.

5.2 Node Embedding

5.2.1 Datasets. We performed a node embedding task on three benchmark graph datasets: Cora, Citeseer, and BlogCatalog. Both Cora and Citeseer are the citation graphs between academic papers, and BlogCatalog is the friendship graph between social network users. Differently to the user feedback dataset, the sparsity of data is again almost low indicates being dense and navy blue being sparse.

Table 5: Summary statistics of the graph datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Nodes</th>
<th># Edges</th>
<th># Classes</th>
<th>Attribute</th>
<th>Sparsity</th>
<th>Skewness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>5,429</td>
<td>7</td>
<td>O</td>
<td>0.999</td>
<td>15.721</td>
</tr>
<tr>
<td>Cite</td>
<td>3,312</td>
<td>4,732</td>
<td>6</td>
<td>O</td>
<td>0.999</td>
<td>9.989</td>
</tr>
<tr>
<td>Blog</td>
<td>10,312</td>
<td>333,983</td>
<td>37</td>
<td>X</td>
<td>0.999</td>
<td>9.823</td>
</tr>
</tbody>
</table>

Figure 7: TSNE visualizations of the learned embeddings from AutoRec and TRAP$^{auto}$ on the ML1M dataset. The color shows the normalized sparsity of each data object, where yellow indicates being dense and navy blue being sparse.

5.2.2 Algorithms.

- **DeepWalk** [19]: A basic random walks-based model that uses the SkipGram embedding architecture.
- **LINE** [24]: A random walks-based model that focuses on preserving the first-order and second-order proximities.
- **Node2Vec** [7]: A random walks-based model that uses breadth-first and depth-first sampling instead of random walk proximity sampling.

---

3. [http://socialcomputing.asu.edu/datasets/BlogCatalog3](http://socialcomputing.asu.edu/datasets/BlogCatalog3)
5.2.4 Evaluation. To measure the performance of the node embedding, we repeated every task five times and reported the average of each metric.

5.2.5 Performance Comparison.

- **Node Classification**: Figures 8 and 9 show the micro-$f_1$ and macro-$f_1$ scores of all algorithms on three datasets with varying ratios of training data in $\{10\%, 30\%, 50\%, 70\%, 90\%\}$. Then, we performed the task using two simple classifiers: OneVsRestClassifier and LogisticRegressor from scikit-learn\(^\text{10}\).

- **Graph Reconstruction**: This task validates how well the learned embeddings preserve the structural information of the graph. For the performance evaluation, we adopted a widely-used metric $\text{precision}_{\text{GR}}@k$, which indicates how many $k$-nearest neighbors retrieved using each learned embedding match the true adjacent nodes in the graph. Given the adjacency matrix $X$ of $M$ users (i.e., the binarized matrix in Section 5.2.1), $\text{precision}_{\text{GR}}@k$ is defined by Eq. (18), where $(i,j)$ is a pair of users. Since DANE uses additional attribute information unlike other embedding methods, it was excluded from the overall comparison for fairness.

$$\text{Precision}_{\text{GR}}@k = \frac{1}{K} \sum_{i,j} |\{\text{rank}(i,j) < k\} \cap \{X(i,j) = 1\}|$$ (18)

In support of reliable evaluation, we repeated every task five times and reported the average of each metric.

[10]:https://scikit-learn.org/stable

---

Figure 8: Micro-$f_1$ scores of six graph embedding algorithms on three datasets.

Figure 9: Macro-$f_1$ scores of six graph embedding algorithms on three datasets.
Table 6: Micro-f1 and macro-f1 of all algorithms on two datasets (the best results are marked in bold).

<table>
<thead>
<tr>
<th></th>
<th>Cora</th>
<th>Cite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio of training data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>30%</td>
</tr>
<tr>
<td>micro-f1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DANE</td>
<td>0.776</td>
<td>0.822</td>
</tr>
<tr>
<td>TRAPDANE</td>
<td>0.793</td>
<td>0.831</td>
</tr>
<tr>
<td>macro-f1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DANE</td>
<td>0.754</td>
<td>0.806</td>
</tr>
<tr>
<td>TRAPDANE</td>
<td>0.774</td>
<td>0.818</td>
</tr>
</tbody>
</table>

Table 7: PrecisionGR@k of six graph embedding algorithms on three datasets (the best results are marked in bold).

<table>
<thead>
<tr>
<th>PrecisionGR@</th>
<th>10</th>
<th>100</th>
<th>500</th>
<th>1,000</th>
<th>10,000</th>
<th>100,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>0.6</td>
<td>0.54</td>
<td>0.518</td>
<td>0.504</td>
<td>0.256</td>
<td>0.055</td>
</tr>
<tr>
<td>LINE</td>
<td>1.7</td>
<td>0.76</td>
<td>0.328</td>
<td>0.219</td>
<td>0.052</td>
<td>0.013</td>
</tr>
<tr>
<td>Node2Vec</td>
<td>0.7</td>
<td>0.5</td>
<td>0.45</td>
<td>0.31</td>
<td>0.066</td>
<td>0.026</td>
</tr>
<tr>
<td>Struct2Vec</td>
<td>0.6</td>
<td>0.36</td>
<td>0.182</td>
<td>0.125</td>
<td>0.047</td>
<td>0.012</td>
</tr>
<tr>
<td>SDNE</td>
<td>0.73</td>
<td>0.638</td>
<td>0.584</td>
<td>0.509</td>
<td>0.305</td>
<td>0.069</td>
</tr>
<tr>
<td>TRAPDANE</td>
<td>1.7</td>
<td>0.858</td>
<td>0.814</td>
<td>0.453</td>
<td>0.099</td>
<td></td>
</tr>
<tr>
<td>DeepWalk</td>
<td>1.0</td>
<td>0.7</td>
<td>0.19</td>
<td>0.154</td>
<td>0.131</td>
<td>0.044</td>
</tr>
<tr>
<td>LINE</td>
<td>1.0</td>
<td>0.68</td>
<td>0.498</td>
<td>0.349</td>
<td>0.068</td>
<td>0.014</td>
</tr>
<tr>
<td>Node2Vec</td>
<td>0.8</td>
<td>0.76</td>
<td>0.438</td>
<td>0.248</td>
<td>0.047</td>
<td>0.020</td>
</tr>
<tr>
<td>Struct2Vec</td>
<td>0.6</td>
<td>0.29</td>
<td>0.14</td>
<td>0.109</td>
<td>0.037</td>
<td>0.012</td>
</tr>
<tr>
<td>SDNE</td>
<td>0.8</td>
<td>0.82</td>
<td>0.584</td>
<td>0.497</td>
<td>0.227</td>
<td>0.045</td>
</tr>
<tr>
<td>TRAPDANE</td>
<td>1.0</td>
<td>0.736</td>
<td>0.672</td>
<td>0.318</td>
<td>0.074</td>
<td></td>
</tr>
<tr>
<td>DeepWalk</td>
<td>1.0</td>
<td>0.92</td>
<td>0.88</td>
<td>0.486</td>
<td>0.152</td>
<td></td>
</tr>
<tr>
<td>LINE</td>
<td>1.0</td>
<td>1.0</td>
<td>0.998</td>
<td>0.669</td>
<td>0.125</td>
<td></td>
</tr>
<tr>
<td>Node2Vec</td>
<td>0.98</td>
<td>0.984</td>
<td>0.910</td>
<td>0.435</td>
<td>0.145</td>
<td></td>
</tr>
<tr>
<td>Struct2Vec</td>
<td>0.75</td>
<td>0.5</td>
<td>0.272</td>
<td>0.217</td>
<td>0.126</td>
<td>0.098</td>
</tr>
<tr>
<td>SDNE</td>
<td>1.0</td>
<td>0.92</td>
<td>0.868</td>
<td>0.830</td>
<td>0.689</td>
<td>0.477</td>
</tr>
<tr>
<td>TRAPDANE</td>
<td>1.0</td>
<td>0.98</td>
<td>0.948</td>
<td>0.826</td>
<td>0.545</td>
<td></td>
</tr>
</tbody>
</table>

attribute information to further improve the embedding performance. Table 6 summarizes the micro-f1 and macro-f1 scores of TRAPDANE and DANE on two datasets. Similarly, TRAPDANE outperformed DANE by alleviating the polarization problem in all cases.

- **Graph Reconstruction**: Table 7 shows precisionGR@k of all algorithms on three datasets with varying k. In all datasets, TRAPSDNE generally achieved the best performance. In particular, as k increased, its relative improvement compared with SDNE gradually increased by up to 43.48% in Cora, 64.44% in Citeseer, and 14.3% in BlogCatalog. That is, our two regularizers allow us to capture more general structural information from the original graph by handling the polarization problem.

6 CONCLUSION

In this paper, we proposed TRAP, a novel meta-approach to address the polarization problem that exists in many real-world scenarios. We showed that the sparsity of data objects severely affected the embeddings and, therefore, suggested the two-level regularizers: (i) the macroscopic regularizer restricts the overall effect of data sparsity, and (ii) the microscopic regularizer finely tunes objects to become closer to correlated objects in the latent embedding space. TRAP can be easily combined with most autoencoder-based approaches by adding the regularizer into the loss function and changing the AE architecture. We validated the effectiveness of our approach on two independent tasks using six datasets, and TRAP always significantly improved the performance when applied to existing embedding methods. Overall, we believe that our work successfully tackled the polarization problem to greatly enhance the learning capability of embedding methods.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (No. 2017R1E1A1A01075927).

REFERENCES


1623


