Privacy of Network Embeddings and the Right to Forget

ACM Reference Format:
. 2022. Privacy of Network Embeddings and the Right to Forget. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Abstract Data ownership and data protection are increasingly important topics with ethical and legal implications. In this light, we investigate network embeddings in the context of the "right to forget". The embeddings are network nodes' representation as low dimensional vectors that capture the information about the neighbourhood. We consider a typical social network scenario with nodes representing users and edges relationships between them. We assume that a network embedding of the nodes has been trained. If a user demands removing his data, it requires the complete deletion of the corresponding network information, particularly the corresponding node and incident edges.

It is possible to undertake retraining of the embedding but this may be potentially expensive computationally. Even so, it has been shown [1] that, after the removal of the node from the network and the deletion of the vector representation of the respective node, the sometimes crucial embedding information about the link structure of the removed node is still encoded in the embedding vectors of remaining nodes. Hence, a privacy attack can be deployed by leveraging information from the remaining network and embedding it to recover information about the removed node's neighbours. The attack is based on two key points: (i) measuring distance changes in network embeddings (ii) a machine learning classifier trained on networks constructed by removing additional nodes. We propose a defence to such an attack that is based on imitating the distance change between successive deletions of a node and predicting the most likely neighbours of the deleted node.

The critical idea of defence is to add noise to the neighbouring nodes (of the deleting nodes) at each iteration so that measuring distance changes will not directly imply the adjacent nodes. In the above defence, we have used a Laplacian differential privacy[28] mechanism and, to maintain the structural properties, we have made use of copula functions.

1 Introduction

Network embeddings, which are network nodes' representation as low-dimensional vectors, are a key technique of many state-of-the-art solutions for a broad range of social network analysis tasks such as link prediction [2] and node classification [3]. Since embedding calculation is often computationally expensive, recalculation after every social network change is not feasible for extensive networks. Simultaneously, data ownership and data protection have become increasingly important topics for social media and social network applications. Previous work has demonstrated that machine learning models can leak information about the training data [4–6]. In addition, network embeddings also open up the potential for residual personal data to remain in a system while explicit data regarding a user has been deleted [1].

This work investigates the possibility of recovering private information from social network embeddings and the measures taken to stop this confidential information leakage. We assume a social network with users as nodes and friendship relations as edges. The low dimensional vector representation of the node is obtained by a standard embedding algorithm (such as [7]). Now, one specific user deletes his account, requests the removal of his data, and thus legally requiring the platform to delete all data of this user. In this setting, it has been observed that removing the node from the network and deletion of the vector of the respective node is not sufficient from a privacy perspective. Significant neighbourhood information of the removed node is still encoded in the embedding vectors of the remaining nodes.

An entity with higher node centrality[1] is more likely to contribute to a greater risk of privacy violation when data is aggregated. While there has been work([8]) on determining the privacy score of a user based on the centrality of the user with or without the information of the complete network, here we have developed an approach that uses differential privacy to come up with a flexible, user-dependent risk framework.

To develop the above-proposed framework, we have used ϵ-differential privacy where flexibility is provided to users by giving the freedom to choose the privacy loss parameter(ϵ); the smaller the ϵ better the privacy protection and vice versa.

2 Background and Related Work

2.1 Network Embeddings

Network embeddings has recently received substantial attention from the research community due to their excellent performance for practical applications such as link prediction, node classification, visualization, clustering, or community detection. Thus, a large range of methods for the calculation of network embeddings has been proposed in literature including factorization based methods [14], random walk-based methods [15], or Convolutional Neural Networks [16]. Although our approach's performance might vary, it is, in principle, agnostic to the algorithms used for training the embedding. Thus, we do not elaborate further on those approaches' specifics but refer to recent surveys on the topic [17, 18].

2.2 Link Prediction

One specific application area of social network analysis is link prediction, i.e., the prediction of missing links between nodes in a network or those that are likely to occur in the future or were removed. Predicting links to a specific node can be used to identify

[1]i.e. connected to a larger number of others

Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

private information in anonymized networks (e.g., [13]). However, in link prediction, typically information on the current network structure and/or on attributes of nodes (for which links are to be predicted) are exploited.

Neither is available in our scenario for the removed node as we assume this information to be removed entirely from the data. Thus, we cannot apply standard link prediction methods such as similarity-based methods or stochastic block models.

2.3 Types of Privacy attacks on machine learning models

Since our work involves using ML to reconstruct a deleted node's information, we briefly discuss relevant privacy attacks on machine learning models.

- Model Inversion Attacks: An adversary tries to infer sensitive attributes' values in the training data given a traditional machine learning model. Generally, attack scenarios can be divided into two variations: White-box attacks ([9]) where the adversary has access to information about the model architecture and/or parameters, and black-box attacks where the adversary does not have access to properties of the model. Black-box attacks are more common and relevant in terms of data privacy. In most scenarios, the adversary can use the model, by applying inputs to the model and observe corresponding outputs (e.g., machine learning offered as a service). Many approaches evaluate different inputs on the model to check how likely they are part of the training data. If the input space is small enough, all possible inputs can be tested; for larger input spaces (domain-specific), gradient descent based approaches can be used [10].
- Membership Inference Attack: These attacks aim to detect if an input for a machine learning model was used during training [5, 11, 12]. The basic idea is to extract differences in the confidence intervals of outputs on data used during training and those not used.
- Train Shadow Models: These attacks aim to imitate the attacked model but on different generated training datasets by training shadow models [29, 30] using the same machine learning platform as was used to train the target model. The training datasets of the target and shadow models have the same format but are disjoint. An attack classifier is trained on data from these shadow models and is applied to the attacked machine learning model's data to achieve the attack's goal. Refer figure 1

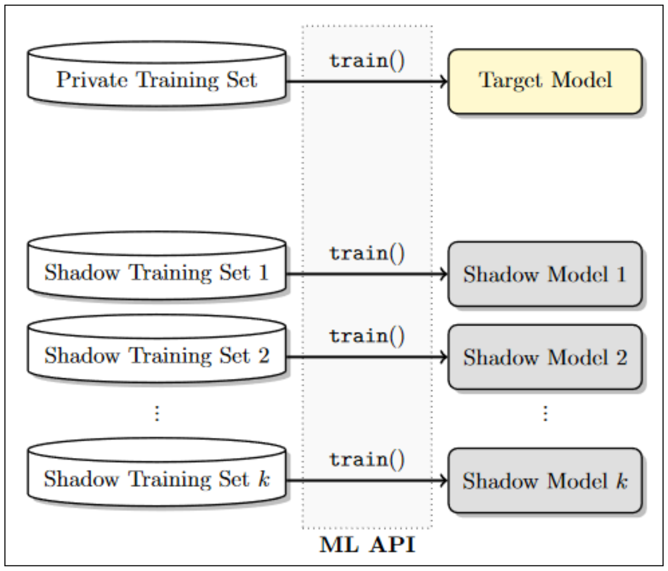

Figure 1: Train Shadow Models

3 Some Technical Concepts

3.1 Differential Privacy

Differential privacy[25] (DP) addresses the paradox of learning nothing about an individual while learning useful information about a population. It specifies privacy using a probabilistic model but does not spell out the exact means of achieving it. For example, DP can be achieved by adding a specific type of randomness into raw data.

Let $|X|$ be the dimensionality of data item X. Then the l_1 norm of dataset x is given by $||x||_1$; mathematically,

$$||x||_1 = \sum_{i=1}^{|X|} |x_i|$$

A randomized algorithm[2] M with domain $N^{|X|}$ is (ϵ, δ)-differentially private if for all $S \subseteq \text{Range}(M)$ and for all $x, y \in N^{|X|}$ such that the distance between the datasets $||x - y||_1 \leq 1$:

$$\Pr[M(x) \in S] \leq exp(\epsilon)\Pr[M(y) \in S] + \delta$$

where

- Epsilon(ϵ): A metric of privacy loss with a differential change in data (adding or removing 1 entry). The smaller its value, the better privacy protection but less accurate result. Value of ϵ should be selected by considering the trade-off between privacy and accuracy.
- delta(δ): δ signifies the probability of information accidentally being leaked.
- y represents the reduced dataset with $n-1$ entries, while total number of entries is n.

3.2 Copula Function

Copula functions[24] are used to describe the dependence between multivariate random vectors and allow us to build the multivariate joint distribution using one-dimensional marginal distributions. Copula functions are a family of distribution functions representing the dependence structure implicit in a multivariate random vector. Intuitively, any high-dimensional data can be modeled as two parts:

- marginal distributions of each individual dimension.
- the dependence among the dimensions.

[2]A randomized algorithm employs a degree of randomness as part of its logic. The algorithm typically uses uniformly random bits as an auxiliary input to guide its behaviour in the hope of achieving good performance in the "average case" over all possible choices of randomness determined by the random bits; thus, either the running time or the output (or both) are random variables. This can be used when presented with a time or memory constraint, and an average case solution is an acceptable output.

Formally, a randomized algorithm M with domain A and discrete range B is associated with a mapping $M : A \implies \Delta(B)$. On input $a \in A$, the algorithm M outputs $M(a) = b$ with probability $(M(a))_b$ for each $b \in B$.

Copula functions have been shown to be effective for modeling high-dimensional joint distributions based on continuous marginal distributions [20–23]. Copulas are useful due to several reasons:

- When we are provided with more marginal distribution information than the joint distribution of all dimensions, they can be used to generate any joint distributions based on known margins and correlations among all dimensions as it so ever happens in many cases that the underlying distribution of the data may be unknown or different from the assumed distribution, especially for data with arbitrary margins and high dimensions.
- Copulas can be used to model non-parametric dependence for random variables

Copula framework can be used to generate high dimensional and large domain synthetic data. It computes a copula function and samples synthetic data from the function that effectively captures the dependence implicit in the high-dimensional datasets. With the copula functions, we can separately consider the margins and the joint dependence structure of the original data instead of modeling the joint distribution of all dimensions as can be seen in figure 2.

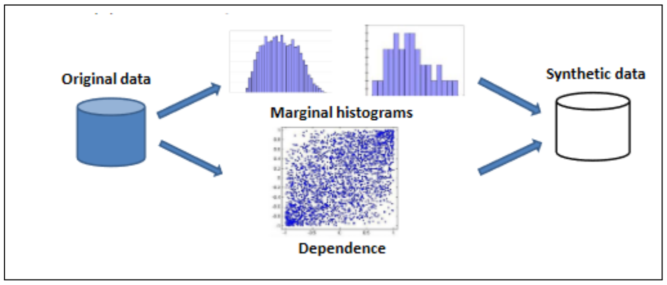

Figure 2: Data generation using copula function

3.3 Hypothesis Testing

Here we recapitulate some basics of hypothesis testing for completeness. In Hypothesis testing in statistics, an analyst tests an assumption regarding a population parameter. It is used to assess the plausibility of a hypothesis by using sample data. Such data may come from a larger population, or from a data-generating process. The word "population" will be used for both of these cases.

There are major four steps involved in an hypothesis test[26]:

- State the hypotheses.
  The hypothesis testing begins with stating the null hypothesis, which is presumed to be true. For example, if we wanted to check whether our data distribution follows some random distribution X, then we will set the null hypothesis that the data set does not follow the distribution X. The null hypothesis (H0), stated as "the null", is a statement about a population parameter, such as population mean that is assumed to be true. The null hypothesis is a starting point of the test whether the null hypothesis is likely to be true though the analyst thinks that it is wrong. Hence, we also define the alternate hypothesis that directly contradicts the null hypothesis.
- Setting the criteria for a decision.
  Data is collected to show that the null hypothesis is not true, based on the likelihood of selecting a sample mean from a population (the likelihood is the criterion). The p value is the probability of obtaining a sample outcome, given that the value stated in the null hypothesis is true. The decision to reject or retain the null hypothesis is called significance. The p value for obtaining a sample outcome is compared to the level of significance for a decision[3]. If the probability of obtaining a given sample mean is less than p assuming the null hypothesis is true, then it can be concluded that the sample selected is too unlikely and so we reject the null hypothesis.
- Compute the test statistic.
  The test statistic is a mathematical formula that allows us to determine the likelihood of obtaining a sample outcome if the null hypothesis is true. The value of the test statistic is used to make a decision regarding the null hypothesis.
- Make a decision.
  The value of the test statistic is used to make a decision about the null hypothesis. The decision is based on the probability of obtaining a sample mean, given that the value stated in the null hypothesis is true. The following two conclusions can be drawn from the test statistics:
  - Reject the null hypothesis. The sample mean is associated with a low probability of occurrence when the null hypothesis is true.
  - Retain the null hypothesis. The sample mean is associated with a high probability of occurrence when the null hypothesis is true.

We now discuss one specific test used in our work; any other suitable method can also be used.

3.3.1 Mann-Whitney U Test The Mann–Whitney U test[27] (also called the Mann–Whitney–Wilcoxon (MWW), Wilcoxon rank-sum test, or Wilcoxon–Mann–Whitney test) is a non parametric test of the null hypothesis for randomly selected values X and Y from two populations that is used to test whether two samples are likely to derive from the same population. It has been said that this test as comparing the medians between the two populations.

The null and alternate hypothesis for non parametric test are as follows:

- null hypothesis(H_0): The two population distributions are equal versus.
- alternate hypothesis(H_1): The two populations are not equal.

Underlying assumptions to use the Mann-Whitney U test are:

- All the observations from both groups are independent of each other.
- The responses are at least ordinal (i.e., one can at least say, of any two observations, which is the greater).
- Under the null hypothesis H_0, the distributions of both populations are equal.
- The alternative hypothesis H_1 is that the distributions are not equal.

[3]The likelihood or level of significance is typically set at 5 percent in behavioral research studies.

The test statistic for the test is denoted as U and is the smaller of U_1 and U_2, defined below.

$$U_1 = n_1 n_2 + n_1 \frac{(n_1+1)}{2} - R_1$$
$$U_2 = n_1 n_2 + n_2 \frac{(n_2+1)}{2} - R_2$$

where n_1 and n_2 are the sample size of both the distributions respectively, and, R_1 and R_2 are sum of the ranks for distribution 1 and 2 respectively.

To determine whether the samples are coming from same distribution, we must determine whether the observed U supports the null or research hypothesis. This is done following the same approach used in parametric testing. Specifically, we determine a critical value of U such that if the observed value of U is less than or equal to the critical value, we reject H_0 in favor of H_1 and if the observed value of U exceeds the critical value we do not reject H_0. If p-value of this test is ≥ significance (say, 0.05), then we are fail to reject the null hypothesis; hence the samples follow same distribution but not if otherwise.

4 A Privacy Attack on Network Embeddings with Node Deletion

Let E be the graph's embedding and $E_{/x}$ be the graph's embedding after removing node x. The embedding algorithm of the original network is assumed to be known to the adversary.

- Start by computing a new embedding E' on the reduced network G' with the same algorithm and parametrization as the original one.
- From this embedding and the available embedding of the original network $E_{/vi}$, calculate respective distance matrices $\Delta_{/vi}$ and Δ' that contains the pairwise distance between each node pair.
- Next, we calculate the changes in distances between these two embeddings by calculating the (element-wise) difference matrix: $\text{Diff}(E_{/vi}, E') = \Delta_{/vi} - \Delta'$. Thus, each element $\text{diff}_{k,l}$ in $\text{Diff}(E_{/vi}, E')$ describes the changes of the distance between nodes v_k and v_l from the original embedding $E_{/vi}$ to the embedding of the remaining network E'.
  Since most embeddings encode in some way a local neighbourhood structure, intuitively nodes that exhibit many or large changes in the embedding distance are more likely to be close to the removed node. But a simple summation of taking maximum values is insufficient for a detailed prediction. Therefore, a machine learning approach identifies which distance changes indicate a node being a neighbour of the removed node.
- For a machine learning approach, a feature vector f_k is constructed for each node $v_k \in G'$ that characterizes the distribution of distance changes for this node in a histogram-like fashion. For that purpose, the range of all values is split in the full matrix $\text{Diff}(E_{/vi}, E')$ (excluding the diagonal) by equal frequency discretization, i.e., bins $b_1, ..., b_m$ are created that contain the same number of values. Now take the distance changes of v_k to all other nodes $\text{diff}_{k,l} | v_l \in G', v_l \neq v_k$. For each bin, the number of these distance changes that lie within the bin then defines one node feature of feature vector f_k. These features are normalized over all bins. As an additional feature, the degree of v_k is also added in G', normalized over all features.
- Next, a training set is created to learn which features and feature combinations predict that a node was a neighbour of the removed node. For that purpose, temporarily remove a second node v_j from G' to obtain a network $G'' = G'_{/v_j}$ and compute the corresponding embedding E''. Then compare the distance changes $\text{Diff}(E'_{/v_j}, E'')$ between the embeddings $E'_{/v_j}$ and E'' and generate node features f for these changes following steps (2)-(4). Additionally, for each node v_k, a label y_k is set that indicates if the respective node was a neighbour of the removed node v_j in G'. To obtain more training data, this process is repeated for each node $v_j \in G'$ as a node to be removed from G'.
- Following that, train a machine learning classifier with the data obtained in the last step, predicting the label y_k (was a node a neighbour of the removed node) based on the features vector f_k (with the distribution of distance changes and the degree) of the node v_k in G''. The intuition is that removing the second node in the network will lead to similar changes in the embedding distances than the changes induced by removing the first node v_i.
- Finally, this classifier on the training features obtained from the network G' and embedding $E_{/v_i}$ to predict which nodes have been neighbouring the removed node in the original network, i.e., which links the removed node had.

4.1 Solution Outline

As the machine learning attack classifier is trained on networks that are constructed by removing additional nodes and the critical component of training data is that it is generated by measuring distance changes in network embeddings. The critical idea of mitigating this issue is if we are able to break the monotonicity of the distance change, i.e. a change in the distance of network embedding is not directly proportional to the removed node/links. To break the monotonicity of the distance change, we propose here the use of a differential privacy mechanism. Informally, we can do this by adding noise to the neighbouring nodes of the deleted nodes. The solution we propose here uses the degree centrality [8] model along with a differential privacy mechanism.

Let $\mathcal{E}$ be the set of embeddings of nodes of graph G at the time of deletion request arrives. As soon as the node deletion is performed, let the deleted node be x, then all the neighbouring nodes of the node x will be added with some amount of noise to prevent the direct implication of neighbouring nodes, or being precise, the hidden information of the deleted data that is still can be recovered from the network by simply analyzing the distance change between nodes(as also mentioned in the attack section that the nodes are directly connected to the deleting node will undergo a major change in distance with other nodes.).

As discussed earlier, Differential privacy (DP) is a model for publicly sharing some summary information about a dataset while withholding information about any specific individual in the dataset. To protect the data privacy of a different potential individual, the DP mechanism adds statistical noise to the data to enhance privacy.

In our proposed approach, we have used laplacian ϵ-differential privacy.

The solution approach follows the following steps.

- Let f be the function that maps the nodes to their corresponding embeddings, i.e. f:v $\implies N^k$, where the node embedding for node v is k-dimensional.
- The laplacian ϵ-differential privacy function on multiple dimensions k is given by:
  $$MLap(x, f(.), \epsilon) = f(x) + (Y_1 + Y_2 + ...Y_k),$$
  where Y_j is sampled from Lap(0, $\frac{\Delta f}{\epsilon}$)
  - Δf refers to the sensitivity of the dataset. Sensitivity is defined as the maximum amount of change seen in a dataset when removing an individual from the dataset.
  - ϵ maintains the trade-off between accuracy and privacy. If ϵ is small, there is higher privacy but data accuracy gets worse. If ϵ is large, privacy will be worse, but data accuracy will be higher. ϵ ranges from 0 to infinity.

We have incorporated a method to compute a user's privacy score based on the degree centrality. In our approach, sensitivity is defined as the degree centrality of the node, with the latter being simply a count of how many social connections (i.e., edges) it has. The straightforward reason for doing this is that the node with a higher degree is more prone to have privacy problems than those with a lower degree; hence, more noise should be added in order to strengthen the privacy guarantee.

As we are adding noise to the data, we need to ensure that the structural or distributional properties of the dataset remain the same. For example, in a simple setting like a graph, the structural invariance can be considered as maintaining the edge distribution. To ensure the structural invariance of the dataset, we have made use of copula functions.

As mentioned in section 2.5, copulas are functions that enable us to separate the marginal distributions from the dependency structure of a given multivariate distribution. They identify the marginal distributions of each attribute and the dependency (in terms of correlation) separately, and then by using this information, they sample the dataset again by using the joint distributions of attributes from their marginal distributions and correlation function. We have considered each dimension of the embedding as a separate marginal distribution in our approach, and the correlation matrix is calculated separately. This process is repeated for both the original embedding set and the one obtained by deleting the node embedding. We generate one sample from each distribution; each such sample is sampled from the synthesized dataset, being the output of the copula function. We check whether these samples follow the original distribution.

To check whether the distribution of the embeddings of the nodes after the deletion of node, say x, is same as the distribution of the complete set of embeddings of the nodes (i.e. without any deletion), we have used the Mann-Whitney-u test mentioned in section 2.7. If the test returns a p-value with p-value > the significance s (say, 0.05), the hypothesis fails. Hence, samples follow the same distribution. Whenever the p-value is less than s, it implies that the null hypothesis is true, i.e. the samples no longer follow the same data distribution. Whenever the differentially private solution data does not follow the desired data distribution, retraining of the network is needed.

4.2 Some Details of the Solution

To simulate an attack scenario, we compute an embedding on the original network G and then remove a node v_i from the network and its corresponding vector from the embedding. The network without the removed node $G_{/vi} = G^{'}$ and the embedding without the removed node $E(G)_{/vi} = E_{/v_i}$ is then given as information for the attack. Results are evaluated using neighbours of v_i in G with the use of 10 bins for feature creation. The embedding dimension is set to 128. The classifier used is Gaussian Naive Bayes classifier with standard parametrizations in the `scikit-learn` python package.

For the mitigation of attack, we have experimented on the embedding generation algorithm on which the attack has been crafted. To incorporate the degree centrality algorithm in calculating the sensitivity parameter of the DP algorithm, we have separately maintained a dictionary so that the degree of the nodes can be easily fetched as all the graphs we have taken for experimentation are undirected; hence, the in-degree is equal to out-degree of nodes. The parameter δ of ϵ DP algorithm is set to 0 with the intention to provide higher privacy guarantees. Mann-Whitney-U test is used from the `scipy-python package`.

For evaluation, we employ the area under curve curve (AUC) of the prediction of neighbors to the removed node, the precision@10, (i.e., the precision among the 10 candidates with the highest predicted likelihood of being a neighbor of the removed node), as well as the F_1-score of the prediction. For the F_1-score, we distinguish between averaging over all attack scenarios (Macro-F_1) and over (to predict) incident edges of the removed nodes in all scenarios (Micro-F_1).

5 Datasets

For larger networks, we use snowball sampling to extract subnetworks with 2000 nodes. All the datasets for privacy attack do not have node attributes. It is just a graph with edges connecting nodes.

- **DBLP:**
  This network is a collaboration network extracted from the DBLP computer science bibliography. Nodes correspond to authors. Two nodes are connected if the two authors collaborated in at least one paper. The used network is snowball sampled. It has 2000 nodes and 7036 edges.
- **Hamstersterː**
  Hamsterster was a social network for people who like hamsters. It operated for 10 years but is shut down now. We use the most significant connected component of this network, which has 1788 nodes and 12476 edges.
- **Facebook:**
  This network is a subset of the friendship network of Facebook. A node represents a user, and an edge between two nodes represents a friendship between the users. A snowball sampled network with 2000 nodes and 14251 edges.

6 Results

6.1 Effectiveness of attack

Table 1 shows the effectiveness of the attack on different networks and algorithms. We can infer that the attack extracts some information from all embeddings on most networks. There is no embedding algorithm the attack clearly performs best on.

Networks		HOPE	LINE	node2vec	SDNE
Facebook	AUC	0.91	0.7	0.72	0.73
Facebook	precision@10	0.6	0.45	0.41	0.42
Facebook	Macro-F_1	0.34	0.33	0.17	0.22
Facebook	Micro-F_1	0.29	0.2	0.2	0.15
Hamsterster	AUC	0.83	0.69	0.66	0.72
Hamsterster	precision@10	0.39	0.29	0.27	0.25
Hamsterster	Macro-F_1	0.19	0.23	0.19	0.18
Hamsterster	Micro -F_1	0.14	0.24	0.22	0.18
DBLP	AUC	0.96	0.74	0.6	0.65
DBLP	precision@10	0.16	0.14	0.08	0.07
DBLP	Macro-F_1	0.19	0.11	0.06	0.08
DBLP	Micro-F_1	0.21	0.12	0.05	0.06

Table 1: Effectiveness of the attack

6.2 Effectiveness of attack after applying DP

Table 2 shows the effectiveness of the attack after employing the DP based solution while table 3 and table 4 shows the effectiveness of the attack for the values of ϵ 0.5 and 0.05 respectively.

Networks		HOPE	LINE	node2vec	SDNE
Facebook	AUC	0.85	0.67	0.68	0.72
Facebook	precision@10	0.59	0.36	0.40	0.39
Facebook	Macro-F_1	0.28	0.30	0.16	0.19
Facebook	Micro-F_1	0.29	0.15	0.2	0.12
Hamster	AUC	0.82	0.68	0.66	0.70
Hamster	precision@10	0.39	0.25	0.23	0.25
Hamster	Macro-F_1	0.18	0.20	0.19	0.14
Hamster	Micro -F_1	0.14	0.24	0.21	0.18
DBLP	AUC	0.88	0.72	0.6	0.6
DBLP	precision@10	0.14	0.14	0.05	0.07
DBLP	Macro-F_1	0.17	0.10	0.06	0.05
DBLP	Micro-F_1	0.20	0.11	0.05	0.05

Table 2: Effectiveness of the attack after DP

6.3 Impact of node degree

Here we study the attack on graphs sampled with different degrees of nodes (low, medium and high) using snowball sampling of graph. Table 5, table 6 and table 7 show the tradeoff of effectiveness of attack, before DP and after applying DP for Facebook, Hamsterster and DBLP datasets, respectively. The entries in these tables are of the form a/b, where 'a' represents the effectiveness of attack (basic) whereas 'b' represents the effectiveness of attack after applying DP.

Networks		HOPE	LINE	node2vec	SDNE
Facebook	AUC	0.90	0.68	0.72	0.72
Facebook	precision@10	0.59	0.37	0.42	0.40
Facebook	Macro-F_1	0.30	0.30	0.16	0.20
Facebook	Micro-F_1	0.25	0.19	0.2	0.13
Hamster	AUC	0.82	0.68	0.66	0.70
Hamster	precision@10	0.39	0.27	0.26	0.25
Hamster	Macro-F_1	0.18	0.21	0.19	0.16
Hamster	Micro -F_1	0.14	0.22	0.21	0.17
DBLP	AUC	0.91	0.72	0.6	0.63
DBLP	precision@10	0.15	0.14	0.07	0.07
DBLP	Macro-F_1	0.16	0.10	0.06	0.05
DBLP	Micro-F_1	0.20	0.12	0.05	0.05

Table 3: Effectiveness of the attack after DP,eps=0.5

Networks		HOPE	LINE	node2vec	SDNE
Facebook	AUC	0.84	0.68	0.72	0.72
Facebook	precision@10	0.59	0.35	0.42	0.39
Facebook	Macro-F_1	0.27	0.30	0.16	0.19
Facebook	Micro-F_1	0.28	0.17	0.2	0.13
Hamster	AUC	0.76	0.65	0.63	0.71
Hamster	precision@10	0.37	0.25	0.26	0.25
Hamster	Macro-F_1	0.18	0.21	0.19	0.14
Hamster	Micro -F_1	0.14	0.23	0.21	0.18
DBLP	AUC	0.87	0.73	0.6	0.62
DBLP	precision@10	0.14	0.14	0.07	0.07
DBLP	Macro-F_1	0.15	0.10	0.06	0.05
DBLP	Micro-F_1	0.20	0.10	0.05	0.05

Table 4: Effectiveness of the attack after DP,eps=0.05

7 Analysis

It is evident from table 2 that the DP mechanism is successful in mitigating the attack to some extent on most of the networks. The high AUC values in the attack scenarios before the DP applied up to 0.96 shows the approach recovers good, partially excellent, candidates for neighbours of the removed node and the reduction in AUC values in the attack scenario after the DP applied up to 0.88 shows that the likelihood of recovering the candidates for the neighbours of the removed has decreased.

As mentioned in the referenced paper for the attack[1] that, in general, there is no embedding algorithm for which the attack performs best over all the networks; similarly, there is no embedding algorithm for which the DP mechanism performs best all over networks because different algorithms vary in behaviour on different

Algo	Metric	low	medium	high
HOPE	AUC	1/0.95	0.86/0.84	0.86/0.80
HOPE	prec@10	0.12/0.10	0.84/0.82	0.84/0.77
HOPE	Macro-F_1	0.03/0.03	0.35/0.35	0.35/0.32
HOPE	Micro-F_1	0.02/0.02	0.36/0.35	0.34/0.34
LINE	AUC	0.7/0.68	0.67/0.65	0.72/0.68
LINE	prec@10	0.08/0.075	0.63/0.62	0.65/0.63
LINE	Macro-F_1	0.05/0.05	0.21/0.21	0.23/0.21
LINE	Micro-F_1	0.04/0.032	0.21/0/19	0.23/0.23
node2vec	AUC	0.72/0.70	0.7/0.65	0.72/0.70
node2vec	prec@10	0/0	0.62/0.60	0.6/0.56
node2vec	Macro-F_1	0/0	0.21/0.21	0.25/0.21
node2vec	Micro-F_1	0/0	0.21/0.20	0.25/0.23
SDNE	AUC	0.77/0.75	0.72/0.71	0.69/0.67
SDNE	prec@10	0.03/0.03	0.63/0.60	0.59/0.57
SDNE	Macro-F_1	0.02/0.02	0.19/0.18	0.19/0.16
SDNE	Micro-F_1	0.01/0.01	0.18/0.18	0.18/0.16

Table 5: Varying node degree for Facebook Dataset

Algo	Metric	low	medium	high
HOPE	AUC	1/1	0.94/0.88	0.95/0.86
HOPE	prec@10	0.06/0.06	0.1/0.1	0.32/0.30
HOPE	Macro-F_1	0.05/0.039	0.21/0.20	0.29/0.26
HOPE	Micro-F_1	0.02/0.02	0.21/0.17	0.32/0.31
LINE	AUC	0.68/0.65	0.76/0.72	0.79/0.73
LINE	prec@10	0/0	0.32/0.30	0.09/0.08
LINE	Macro-F_1	0.01/0.01	0.19/0.18	0.1/0.1
LINE	Micro-F_1	0.01/0.01	0.19/0.16	0.11/0.10
node2vec	AUC	0.6/0.52	0.59/0.55	0.61/0.59
node2vec	prec@10	0.02/0.02	0.14/0.13	0.07/0.05
node2vec	Macro-F_1	0.01/0.01	0.1/0.98	0.06/0.06
node2vec	Micro-F_1	0.01/0.01	0.1/0.1	0.05/0.05
SDNE	AUC	0.6/0.5	0.71/0.70	0.65/0.60
SDNE	prec@10	0/0	0.13/0.11	0.06/0.05
SDNE	Macro-F_1	0.02/0.02	0.09/0.08	0.08/0.08
SDNE	Micro-F_1	0.02/0.017	0.07/0.065	0.08/0.08

Table 7: Varying node degree for DBLP Dataset

Algo	Metric	low	medium	high
HOPE	AUC	1/0.98	0.72/0.70	0.78/0.72
HOPE	prec@10	0.1/0.1	0.57/0.52	0.5/0.43
HOPE	Macro-F_1	0.02/0.02	0.15/0.15	0.19/0.18
HOPE	Micro-F_1	0.02/0.02	0.15/0.13	0.18/0.12
LINE	AUC	0.61/0.59	0.75/0.72	0.72/0.65
LINE	prec@10	0.04/0.037	0.5/0.46	0.34/0.29
LINE	Macro-F_1	0.01/0.01	0.31/0.29	0.27/0.24
LINE	Micro-F_1	0.01/0.01	0.3/0.28	0.27/0.26
node2vec	AUC	0.51/0.43	0.75/0.73	0.73/0.69
node2vec	prec@10	0/0	0.49/0.48	0.33/0.33
node2vec	Macro-F_1	0/0	0.29/0.24	0.29/0.27
node2vec	Micro-F_1	0/0	0.28/0.28	0.28/0.28
SDNE	AUC	0.66/0.64	0.76/0.75	0.72/0.65
SDNE	prec@10	0/0	0.42/0.40	0.32/0.30
SDNE	Macro-F_1	0/0	0.23/0.22	0.21/0.17
SDNE	Micro-F_1	0/0	0.23/0.23	0.21/0.20

Table 6: Varying node degree for Hamsterster Dataset

network structures. The algorithms which can be seen as more unstable[4] in their computations than others showing comparatively better results on DP because the instability is already creating problems for the consistent comparison between embeddings and more over, informally, DP mechanism also introduces uncertainty. Hence, DP is seen to be more beneficial for algorithms that are less stable in their computations.

Node degree variation: It can seen in table 5, table 6, table 7 that all the metric parameters - AUC, prec@10, Macro-F_1 and Micro-F_1 - are quite low for all the embedding algorithms and network for both (with and without DP). Thus, exactly predicting neighbours of low degree nodes as well as mitigation of prediction neighbours of low degree nodes both are very challenging because low degree nodes usually have a smaller impact on the vector representations of their neighbours. AUC varying between very high to low values may be because of a small number of positives. In contrast, the attack achieves a better performance for the nodes with medium and high degree. After DP, it is evident from the results that the nodes with the higher degree show a substantial decrease in performance as compared to the medium degree ones; the possible reason may be the fixed size of vector representations since an increased amount of information must be captured in the same number of dimensions. Thus, as it is already challenging to make a precise prediction of neighbours' of the removed node and our approach adds more uncertainties, it becomes more challenging to predict the neighbourhood of the removed node.

8 Conclusions

When a user requests removal of his data, we have studied in this paper whether a vector deletion that corresponds to a low dimensional representation of a node would suffice the privacy constraints or not. Based on the privacy attack described in [1], it is clear that only vector deletion without complete retraining of network will lead to the information leakage by applying some machine learning algorithm using a trained a classifier.

To provide a defence to this privacy attack, we have proposed the use of the ϵ differential privacy model in combination with node centrality[8]. Copulas play an important role in maintaining the structural invariance in the context of differential privacy. Future work will study graph nodes with multiple attributes.

References

[1] Michael Ellers1, Michael Cochez2, Tobias Schumacher1, Markus Strohmaier1,3, Florian Lemmerich1 1RWTH Aachen University 2VU Amsterdam 3 *Privacy Attacks on Network Embeddings.* https://arxiv.org/abs/1912.10979.

[4]instability of an embedding algorithm can be defined as the outcome of an embedding algorithm that is not deterministic but varies to some degree even if the same algorithm is used on the exact same data because most state-of-the-art embedding algorithms are based on approximate solutions to an optimization function (e.g., via gradient descent).

[2] Liben-Nowell, D., and Kleinberg, J. The link-prediction problem for social networks. *Journal of the American society for information science and technology 58, 7 (2007), 1019–1031.*

[3] Bhagat, S., Cormode, G., and Muthukrishnan, S. *Node classification in social networks. In Social network data analytics. Springer, 2011, pp. 115–148.*

[4] Papernot, N., McDaniel, P., Sinha, A., and Wellman, M. P. Sok. *Security and privacy in machine learning. In European Symposium on Security and Privacy (EuroS & P) (2018), pp. 399–414.*

[5] Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. Mlleaks *Model and data independent membership inference attacks and defenses on machine learning models. arXiv preprint arXiv:1806.01246 (2018).*

[6] Veale, M., Binns, R., and Edwards, L. Algorithms that remember. *model inversion attacks and data protection law. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 376, 2133 (2018), 20180083.*

[7] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. *In International conference on knowledge discovery and data mining (2016), pp. 855–864.*

[8] Varsha Bhat Kukkala and S.R.S Iyengar. *"Identifying Influential Spreaders in a Social Network (While Preserving Privacy)", PETS 2020.*

[9] Ateniese, G., Felici, G., Mancini, L. V., Spognardi, A., Villani, A., and Vitali,D. *Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. arXiv preprint arXiv:1306.4447 (2013).*

[10] Fredrikson, M., Jha, S., and Ristenpart, T. *Model inversion attacks that exploit confidence information and basic countermeasures. In Conference on Computer and Communications Security (2015), pp. 1322–1333.*

[11] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. *Membership inference attacks against machine learning models. In Symposium on Security and Privacy (2017), IEEE, pp. 3–18*

[12] Truex, S., Liu, L., Gursoy, M. E., Yu, L., and Wei, W. *Towards demystifying membership inference attacks. arXiv preprint arXiv:1807.09173 (2018).*

[13] Cai, Z., He, Z., Guan, X., and Li, Y. *Collective data-sanitization for preventing sensitive information inference attacks in social networks. IEEE Transactions on Dependable and Secure Computing 15, 4 (2016), 577–590.*

[14] Balasubramanian, M., and Schwartz, E. L. *The isomap algorithm and topological stability. Science 295, 5552 (2002), 7.*

[15] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. *In International conference on knowledge discovery and data mining (2016), pp. 855–864.*

[16] Kipf, T. N., and Welling, M. *Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).*

[17] Cai, H., Zheng, V.W., and Chang, K. C.-C. *A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Transactions on Knowledge and Data Engineering 30, 9 (2018), 1616–1637.*

[18] Goyal, P., and Ferrara, E. *Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems 151 (2018), 78–94.*

[19] Reza Shokri, Marco Stronati, Cong zheng, Vitaly Shmatikov. *Membership Inference Attacks Agains tMachine Learning Models.*

[20] Bluhm, C OL, Wagner C. *An introduction to credit risk modeling. Chapman and Hall, Boca Raton.*

[21] Barak B, Chaudhuri K, Dwork C, Kale S, McSherry F, Talwar K. *Privacy, accuracy, and consistency too: a holistic solution to contingency table release. PODS. 2007*

[22] Nelsen RB. *An Introduction to Copulas. Springer; 1999.*

[23] Rosenberg J. *Npon-parametric pricing of multivariate contingent claims. Journal of Derivatives.*

[24] Haoran Li, Li Xiong, Xiaoqian Jiang. *Differentially Private Synthesization of Multi-DimensionalData using Copula Functions;2014.*

[25] Cynthia Dwork, Aaron Roth. *The Algorithmic foundations of differential privacy; 2014.*

[26] Daniel Goldman. *The Basics of Hypothesis Tests and Their Interpretations; 2018.*

[27] Nadim Nachar. *The Mann-WhitneyU: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution; 2008.*

[28] Rathindra Sarathy, Krish Muralidhar. *Differential Privacy for Numeric Data; 2009.*

[29] Reza Shokri, Marco Stronati, Congzheng Song, Vitaly Shmatikov. *Membership Inference Attacks AgainstMachine Learning Models;Cornell,2014.*

[30] Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. Mlleaks *Model and data independent membership inference attacks and defenses on machine learning models. arXiv preprint arXiv:1806.01246 (2018).*