Robust Negative Sampling for Network Embedding

Mohammadreza Armandpour,[1] **Patrick Ding,**[1] **Jianhua Huang,**[1] **Xia Hu**[2]
[1]Department of Statistics, Texas A&M University
[2]Department of Computer Science and Engineering, Texas A&M University
{armand, patrickding, jianhua}@stat.tamu.edu, hu@cse.tamu.edu

Abstract

Many recent network embedding algorithms use negative sampling (NS) to approximate a variant of the computationally expensive Skip-Gram neural network architecture (SGA) objective. In this paper, we provide theoretical arguments that reveal how NS can fail to properly estimate the SGA objective, and why it is not a suitable candidate for the network embedding problem as a distinct objective. We show NS can learn undesirable embeddings, as the result of the "Popular Neighbor Problem." We use the theory to develop a new method "R-NS" that alleviates the problems of NS by using a more intelligent negative sampling scheme and careful penalization of the embeddings. R-NS is scalable to large-scale networks, and we empirically demonstrate the superiority of R-NS over NS for multi-label classification on a variety of real-world networks including social networks and language networks.

Introduction

Network embedding aims to reflect the semantic similarity of network nodes by learning a low dimensional latent representation for each node which assigns similar nodes closer embeddings (Cui et al. 2018). Learning the embeddings can be achieved in supervised fashion (Yang, Cohen, and Salakhutdinov 2016; Tu et al. 2016; Huang, Li, and Hu 2017b) or via unsupervised methods (Cao, Lu, and Xu 2016; Lai et al. 2017; Tu et al. 2018; Zhang et al. 2018). The embeddings can be used to facilitate tasks such as structural discovery (Ribeiro, Saverese, and Figueiredo 2017), node classification (Bhagat, Cormode, and Muthukrishnan 2011), community detection (Wang et al. 2017), link prediction (Liben-Nowell and Kleinberg 2003), temporal prediction (Sadeghian et al. 2016), and network visualization (Maaten and Hinton 2008; Shaw and Jebara 2009). A common practice to embed similar nodes close to each other is to define a set of neighbors for each node. These neighborhoods capture similarity between nodes and translate the problem to embedding neighbors close to each other. Some proposed neighborhood definitions for a node include the immediate neighbors as in LINE (Tang et al. 2015); the result of a random walk on the graph as in DeepWalk and Node2Vec (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016); and the result of a random walk on a multi-layer graph in Struc2Vec (Ribeiro, Saverese, and Figueiredo 2017).

To assign neighbor nodes close embedding vectors, many existing network embedding algorithms optimize a variation of the Skip-gram one hidden layer neural net objective (Mikolov et al. 2013a), which originated in the natural language processing (NLP) literature. Due to the computational complexity of the objective these methods rely on NS (Mikolov et al. 2013b) to approximate the objective. Hierarchical Softmax (Morin and Bengio 2005) and Noise Contrastive Estimation (Gutmann and Hyvärinen 2012; Mnih and Teh 2012) are two alternatives to NS, but NS is preferred to its alternatives because of its better performance in most cases (Mikolov et al. 2013b; Tang et al. 2015; Grover and Leskovec 2016; Ribeiro, Saverese, and Figueiredo 2017).

The SGA objective consists of softmax terms and NS approximates each term individually. Each softmax term in the SGA objective is related to a node and one of its neighbors, which encourages the node and the neighbor be close to each other and far from all other nodes. There can be many other nodes, so in each iteration of stochastic gradient descent for optimizing the objective NS instead takes a small random sample of nodes, the negative sample, where high degree nodes are more likely to be chosen. Then the node only needs to be far from the members of the negative sample.

NS provides a reasonable estimate for each term in the SGA objective, but we show the accumulative sum of errors in the estimated terms can lead to embeddings which do not satisfy the desired properties encouraged by the SGA objective. We also discuss why NS objective is not a proper objective for network embedding as a separated objective. Deriving a vertex-level formula for the whole NS objective allows us to demonstrate theoretically why this problem happens. This phenomenon can cause poor embedding of high degree nodes, which leads to low-quality embeddings overall. We refer to this as the "Popular Neighbor Problem" and explain and analyze its effect in more detail.

This shortcoming of NS motivated us to propose R-NS. R-NS uses an adaptive negative sampler without changing the complexity of the algorithm and has a careful design for norm penalization of the embedding with predetermined penalty coefficient. We prove R-NS avoids the Popular Neighbor Problem and illustrate empirically the superiority of R-NS over NS by comparing the effectiveness of the learned embed-

dings for multi-label classification tasks on several large-scale networks such as BlogCatalog, Flickr, and Wikipedia.

There have been some recent attempts to improve NS for NLP applications (Chen et al. 2018; Mu, Yang, and Yan 2018). Chen et al. proposes a more complex way of sampling negative and positive samples which can increase the time complexity of the algorithm based on the precision of the estimation. Their method needs to rank all the words in the vocabulary every few iterations of stochastic gradient descent, which is an expensive procedure. Mu, Yang, and Yan does not change the sampling procedure of NS. They add a penalty on the norm of the embedding to the objective without changing the sampling procedure of NS. Therefore their method still suffers from the Popular Neighbor Problem. It also cannot be applied to unsupervised problems because they lack guidelines for choosing the penalty coefficient.

We also consider some naive solutions to the Popular Neighbor Problem, apply them to large-scale networks, and explain their poor performance using Bayesian and regularization arguments. Our contributions can be summarized as:

- Recognition and analysis of the problems of NS for estimation of the SGA objective through theory and experiments on real network data.
- Proposing R-NS and showing theoretically it alleviates the problem of NS, with a scalable sampling procedure and a careful design of norm penalization with deterministic algorithm for setting the hyperparameters.
- Empirically demonstrating the superior performance of R-NS compared to NS through experiments on a variety of large-scale networks.
- Suggesting other possible alternatives to NS and providing intuition as to why R-NS outperforms them using a Bayesian argument.

Preliminaries

Skip-Gram Architecture for Networks

The Skip-gram architecture (SGA) is a one hidden layer neural network originally introduced for the word embedding problem, which has been successfully extended to the network embedding problem.

In the network context, SGA learns for each node u two d dimensional vectors, an embedding vector $f(u)$ and a context vector $f'(u)$, by maximizing the log probability of observed network neighborhoods of the nodes:

$$\max_{f,f'} \sum_{u \in V} \sum_{n_i \in N(u)} \log P(n_i|u; f, f'),$$

where $N(u)$ is the neighborhood of u and V is the set of all nodes. It models $P(n_i|u; f, f')$ by a softmax unit:

$$P(n_i|u; f, f') = \frac{\exp(f'(n_i)^T \cdot f(u))}{\sum_{v \in V} \exp(f'(v)^T \cdot f(u))},$$

This leads to the objective:

$$\sum_{u} \sum_{n_i \in N(u)} \log \left(\frac{\exp(f'(n_i)^T \cdot f(u))}{\sum_{v \in V} \exp(f'(v)^T \cdot f(u))} \right). \quad (1)$$

We refer to (1) as the Type II objective, and we use the term Type I objective for the case where $f'(u)$ is the same as $f(u)$.

The network embedding algorithms that use SGA differ only in their definition of the neighborhood $N(u)$ and in their usage of Type I or type II objective. For instance LINE defines $N(u)$ as the immediate neighbors of u and uses both Type I (LINE 1st) and Type II (LINE 2nd) objectives. DeepWalk and Node2vec use the Type II objective and a random walk on the graph to specify $N(u)$ (the former employs a uniform random walk while the latter uses a biased random walk). Other methods like Struc2vec first calculate a measure of structural similarity between the nodes and use that to define the neighborhood.

Negative Sampling

The SGA objective is computationally infeasible for large-scale networks because each of the softmax terms requires summation over all vertices. Negative sampling (NS) provides a computationally feasible approximation to the objective by replacing each $\log P(n_i|u; f, f')$ term with

$$\log(\sigma(f'(n_i)^T \cdot f(u))) + \sum_{j=1}^{k} \mathbb{E}_{v_j \sim P(v)} \log(\sigma(-f'(v_j)^T \cdot f(u))), \quad (2)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ and $P(v)$ is a distribution proportional to d_v^β, where d_v is the degree of the node v. The degree power β is a hyper-parameter, conventionally set to 3/4 because of good empirical performance(Mikolov et al. 2013b). The objective can be efficiently optimized with mini-batch gradient descent or stochastic gradient descent (SGD) because the objective is a summation of terms like (2) and each term (for example term related to (u, n_i), for $n_i \in N(u)$) can be estimated by:

$$\log(\sigma(f'(n_i)^T \cdot f(u))) + \sum_{j=1}^{k} \log(\sigma(-f'(v_j)^T \cdot f(u))), \quad (3)$$

The expectation of (3) is equal to (2). The v_j are the negative neighbor sample for u and are drawn from the distribution $P(v)$. In the Type I objective case $f' = f$ in (3) and (2). In subsequent sections we present theorems or claims for the type II objective setting, but in most cases they apply for the type I objective setting without modification.

Robust Negative Sampling

Here we show NS is hampered by a problem that we call the "Popular Neighbor Problem." We explain how the problem arises in NS, and how it affects the learned embeddings. We propose R-NS to alleviate the problem. It uses a scalable adaptive negative sampler and penalization of the norm of the embedding with a carefully designed equation for the penalty coefficient based on the properties of the network.

The Popular Neighbor Problem

A good embedding method, or equivalently a good objective, is one which encourages neighboring nodes to have similar embedding vectors. We will show that the NS objective does not completely promote this behavior. Therefore it is not an ideal candidate for an embedding method, neither as an approximation to the SGA objective nor on its own merits as an objective. This deviation of NS from that ideal behaviour is mainly caused by allowing a node to choose its neighbor as a negative sample. This problem is more severe when high-degree nodes are present. We refer to this as the Popular Neighbor Problem to emphasize that the situation is worse for a node with high degree neighbors.

To be more specific, the Popular Neighbor Problem can cause over-shrinkage of embeddings of high degree nodes and large angular distance between embeddings of neighboring nodes under the Type I objective, and large angular distance between the embedding of a node and context vector of its neighbor under the Type II objective, which are contradictory to what we ideally expect from an embedding method.

In the following, we support the above claim theoretically. We also show that the SGA objective supports the behaviour of embedding neighbors close to each other. We start by deriving the objective of SGA and NS at the vertex-level. The vertex-level objective is the summation of all the $\log P(n_i|u; f, f')$ terms for all $n_i \in N(u)$, therefore it contains the contribution of all the neighbor nodes. The vertex-level objective is more desirable and meaningful to look at compared to the edge-level objective because it contains the effect of all other nodes on a specific node.

Theorem 1. *Let $Z_{(v_j,u)} = f'(v_j)^T \cdot f(u)$ The vertex level type II objective for SGA is:*

$$-d_u \log\left(\sum_{v_i \in V} e^{Z_{(v_i,u)}}\right) + \sum_{v \in N(u)} Z_{(v,u)}, \qquad (4)$$

while for NS it is:

$$\sum_{v_i \in V} -\left(\frac{d_{v_i}^{\beta} k d_u}{C} + \mathbb{1}_{v_i \in N(u)}\right) \log(1 + e^{Z_{(v_i,u)}}) + \sum_{v \in N(u)} Z_{(v,u)}, \qquad (5)$$

where $C = \sum_{v_j \in V} d_{v_j}^{\beta}$. In the type I case $f' = f$.

Proof. Proof provided in the supplementary material. □

To have a better intuition about the objectives, note that $Z_{(v,u)}$ is a measure of the similarity between $f'(v)$ and $f(u)$ and the right terms in both (4) and (5) are increasing functions of $Z_{(v,u)}$. Therefore the right terms of the objectives encourages a node to be similar to its neighbors. However, the left terms in both objectives encourage u to be dissimilar to all other nodes because those terms have the form $-a \log(e^{Z_{(v,u)}} + T)$ where $a, T > 0$ and that function is decreasing with respect to $Z_{(v,u)}$. For convenience we refer to the left terms in (4) and (5) as L_u^{SGA} and L_u^{NS}, respectively.

The following theorem shows that NS does not promote similarity between a node and its neighbors. In fact it puts an upper limit for the similarity while SGA objective does not have such an upper bound. That property of NS is in conflict with both the aim of embedding methods in general, and the SGA objective in particular. Therefore NS is neither a good candidate for approximation of SGA nor for an embedding algorithm in general.

Theorem 2. *If we consider objectives* (4) *and* (5) *as functions of $Z_{(v,u)}$ without the constraint that $Z_{(v,u)}$ is inner product of $f'(v)$, $f(u)$, then the optimal solutions have closed forms. In SGA:*

$$Z_{(n_i,u)} = +\infty \text{ for } n_i \in N(u)$$
$$Z_{(r,u)} = -\infty, \text{ for } r \notin N(u).$$

In NS:

$$Z_{(n_i,u)} = -\log\left(\frac{d_{n_i}^{\beta} k d_u}{C}\right), \text{ for } n_i \in N(u)$$
$$Z_{(r,u)} = -\infty, \text{ for } r \notin N(u).$$

Proof. The proof for the SGA objective is based on the fact that optimization for Z values can be seen as minimization of the KL-divergence between a uniform distribution on the neighbors of the node and the probability distribution on all nodes induced by the softmax unit in the SGA. For NS the optimal values have been calculated by setting the derivative of the vertex-level objective of theorem 1 equal to zero. For the complete argument please see the supplementary material. □

The above theorem shows NS does not allow the similarity of a node and its neighbor to be larger than $-\log\left(\frac{d_{n_i}^{\beta} k d_u}{C}\right)$ (which can be considered upper-bound). In contrast SGA tries to have as much similarity as possible between a node and its neighbor. Note that the upper-bound of the similarity can be very small when either the degree of the neighbor or β is large, which supports our claim that the presence of high-degree neighbors can make the situation worse in NS. This is a serious problem because scale-free networks, which are an important class of interesting networks, include at least a few nodes with high degree nodes.

Notice that even for the case where $\beta = 0$, or for the case where the network does not have high degree nodes, the upper-bound is still not $+\infty$. This means NS still does not encourage similarity between neighbor nodes, only to a lesser degree. The other point is that in Theorem 2 we considered $Z_{n_i,u}$ as an independent variable so it is not valid to try to compare the ideal similarity of two neighbors of node u together for n_i, n_j.

Positive but non-infinite optimal similarity can still lead to over-shrinkage because the embedding vector of high degree nodes needs to become very small to make the inner product of its vector and its neighbors' vectors small but positive. This over-shrinkage causes low-quality embedding and difficulty distinguishing among high-degree nodes.

To empirically illustrate the effect of the Popular Neighbor Problem we embedded the Zachary's karate club network

(Zachary 1977) in $\mathbb{R}^2$ in order to visualize the embeddings directly, without any dimension reduction. The results support the above claims and demonstrate over-shrinkage of high degree nodes' embedding vectors. The results are provided in the supplementary material. The neighborhood definition is given in the first part of the experiment section.

Distant Sampling and Explicit Norm Penalization

To provide insight about our proposed solution we analyze both the SGA and NS objectives in more detail, and show how they implicitly penalize the norm of the embedding vectors. We discuss the necessity of norm penalization and explain R-NS precisely. We conclude by proving theoretically that R-NS does not suffer from the Popular Neighbor Problem.

The terms L_u^{SGA}, and L_u^{NS}, defined in the previous subsection, implicitly penalize the norm of node u's embedding vector. The presence of terms like $\exp(Z_{(n_i,u)})$ for $n_i \in N(u)$ in L_u^{SGA} and L_u^{NS} cause the norm penalization because of the small angular distance between $f(u)$ and $f(n_i)$ ($f'(n_i)$) in the type I (type II) objective setting, which is a desirable property for an embedding procedure.

To make this argument more rigorous we consider the exclusive effect of the term $\exp(Z_{(v,u)})$ in L_u^{SGA} and L_u^{NS} on the embedding and context vectors in the following theorem. To isolate the effect of the $\exp(Z_{(v,u)})$ we treat the other terms inside L_u^{SGA} as a constant M, where M is equal to one for L_u^{NS}, and we compare the norm of the embedding and context vectors before and after the updating procedure of SGD.

The following theorem shows that if the angular distance between two vectors related to the nodes u and v is smaller than $\frac{\pi}{2}$ which is a desirable property to have if v and u are neighbors of each other–then the L_u term in the objective encourages shrinkage of the vectors by making the vectors smaller after each update of the SGD algorithm.

Theorem 3. *Suppose ${f'_u}^{(t)}, {f_v}^{(t)}$ are two vectors in R^d at step t, and define:*

$$F(x,y) := -\log(M + e^{x^T.y}),$$
$${f'_v}^{(t+1)} := {f'_v}^{(t)} + \eta\nabla_1 F({f'_v}^{(t)}, {f_u}^{(t)}),$$
$${f_u}^{(t+1)} := {f_u}^{(t)} + \eta\nabla_2 F({f'_v}^{(t)}, {f_u}^{(t)}),$$

where ∇_1, ∇_2 are gradients with respect to the first and second arguments of F respectively. Then there exists a η_0 where for any step size η satisfying $0 < \eta < \eta_0$:

$$\text{if } |\alpha^t| < \frac{\pi}{2}:$$
$$||f_u^{(t+1)}||_2 < ||f_u^{(t)}||_2, \quad ||{f'_v}^{(t+1)}||_2 < ||{f'_v}^{(t)}||_2,$$
$$\text{if } |\alpha^t| \geq \frac{\pi}{2}:$$
$$||f_u^{(t+1)}||_2 \geq ||f_u^{(t)}||_2, \quad ||{f'_v}^{(t+1)}||_2 \geq ||{f'_v}^{(t)}||_2,$$

where $\alpha^{(r)}$ is the angle between the vectors $f_u^{(r)}$, ${f'_v}^{(r)}$ and ranges from $[-\pi, \pi]$.

Proof. Proof provided in the supplementary material. □

The above theorem also reveals that if the angular distance between u and v is larger than $\frac{\pi}{2}$–which is a sensible property for a node and its non-neighbor–then the L_u term encourages larger norms for the embeddings. We take advantage of this observation in the design of R-NS.

In fact the norm penalization is an advantage of the NS and SGA objectives because it can decrease the degrees of freedom of the model and provide a more stable solution. Since in the network embedding problem we need to estimate a huge number of parameters, equal to the number of nodes in the network multiplied by the dimension of the embedding, implicit or explicit norm penalization can be helpful, acting as a soft way of reducing the degrees of freedom of the whole model and prevents over-fitting. The concept of effective degrees of freedom and the effect of penalization on the degrees of freedom has been discussed in more detail in (Friedman, Hastie, and Tibshirani 2001).

Now we are ready to explain R-NS and how it can ameliorate the popular neighbor problem. Like NS, R-NS estimates each softmax term in the SGA objective by taking negative samples, but differs in two ways:

- **Distance negative sampler:** It chooses the negative neighbor sample for each node u from the set $V - \{N(u) \cup u\}$ rather than all possible nodes V. To be more specific, R-NS draws a node independently from $P_u(v) \propto d_v^\beta$ for $v \in V - \{N(u) \cup u\}$ but the probability of choosing $v \in \{N(u) \cup u\}$ is set to be zero. The sub-index u in P_u emphasizes that the probability distribution depends on the node. We propose two methods in the following which enable us to draw negative samples from P_u without changing the scalability of the algorithm.
- **Penalty on the norm:** It puts an l_2 penalty on the norm of the embedding vectors for the Type I objective and the embedding and context vectors for the Type II objective. The novelty here is we present a formula for the penalty coefficient λ and it enables R-NS still be applicable to a wide class of unsupervised embedding problems. The formula for λ is calculated based on the careful estimation of the derivative of the whole objective of R-NS and it involves the number of edges, the average weight of edges, the number of nodes, and the dimension of the embedding space. The formula is presented in the following but the exact derivation is presented in the supplementary material.

Based on these two characteristics of R-NS we approximate each $\log P(n_i|u; f, f')$ term in the SGA objective as:

$$\sum_{j=1}^{k} E_{v_j \sim P_u(v)}[\log(\sigma(-{f'(v_j)}^T \cdot f(u))) -$$
$$\frac{\lambda}{k+1}||f'(v_j)||_2] + \log(\sigma({f'(n_i)}^T \cdot f(u))) - \lambda||f(u)||_2$$

If we want to use SGD to optimize the objective we just need to differentiate the following expression at each step:

$$\sum_{j=1}^{k}[\log(\sigma(-{f'(v_j)}^T \cdot f(u))) - \frac{\lambda}{k+1}||f'(v_j)||_2] +$$
$$\log(\sigma({f'(n_i)}^T \cdot f(u))) - \lambda||f(u)||_2 \quad (6)$$

where v_j is drawn from P_u.

Notice that nodes with higher degree have been penalized more in the R-NS objective because they appear more often in the logarithm terms. Therefore, each vector penalized proportional to the number of times it appeared.

In the following we present the formula for λ and provide a brief explanation of its terms. For the type I case:

$$\lambda^{typeI} := \frac{\frac{m}{1+e^{l^2 m}} + k\frac{\sqrt{1-m^2}}{1+e^{l^2\sqrt{1-m^2}}}}{(\frac{k+2}{k+1} + \frac{k}{k+1}\frac{n-2}{n-d-1})}, \tag{7}$$

where $m = \frac{ln(1+\frac{d}{2\bar{w}})}{ln(1+n)}$, $d = \frac{2E}{n}$, $l = \frac{\log_2(\text{dim})}{6} + 2$, E is the number of neighborhood edges, $\bar{w}$ is the mean of the non-zero weights of the edges, dim is the dimension of the embedding space, and n is number of nodes in the graph. The term m acts like the cosine similarity of a node and its neighbor and consequently the term $l^2 m$ acts like the inner product of an embedding of a node and its neighbor. Also notice that for denser networks the term d is larger, and consequently m is larger. For more detail and exact derivation please look at the supplementary file.

For the type II objective case we provide the following approximation for the λ:

$$\lambda^{typeII} := \frac{\frac{m}{1+e^{l^2 m}} + k\frac{\sqrt{1-m^2}}{1+e^{l^2\sqrt{1-m^2}}}}{2}, \tag{8}$$

where l for the type II objective is $\frac{\log_2(\text{dim})}{6} + 1.9$ but all the other notation is same as in the type I case. The only part in the formulation of λ that is calculated experimentally is the functionality of l based on the dimension of the embedding space, however since we fixed the l across all the experiments, the problem still has been solved in unsupervised fashion. Note that although we fixed the l for different networks the λs are different because of the presented formula.

To provide intuition about the two parts of R-NS, the distance negative sampler alleviates the Popular Neighbor Problem and makes R-NS satisfy the desirable properties encouraged by the SGA objective (this fact is shown in the following Theorem 4). However by forcing the negative samples to be from the non-neighbors we lose the implicit norm penalization of the vectors based on the Theorem 3. The explicit norm penalization is designed to compensate for the missing norm penalization and provides a robust solution.

Theorem 4. *The optimal solution for the R-NS objective* (6) *under the same assumptions as Theorem 2 is:*

$$Z_{(v,u)} = \infty, \textit{ for } v \in N(u), Z_{(v,u)} = -\infty, \textit{ for } v \notin N(u).$$

Proof. Proof provided in the supplementary material. □

Theorem (4) shows that R-NS does not suffer from the Popular Neighbor Problem because the optimal $Z_{(v,u)}$ is $+\infty$, which means R-NS does not put an upper bound on the similarity of a node and its neighbor. Moreover by having optimal similarity of $-\infty$ for non-neighbors it promotes dissimilarity between a node and its non-neighbors, which is exactly aligned with the goals of network embedding algorithms. Notice that the penalization of the vectors acts like the logarithm of a normal prior over the embedding vector while the part of (6) related to sampling can be viewed as the log likelihood of the model. Therefore the whole objective of R-NS can be regarded as the logarithm of the posterior distribution of the embedding vectors given the network. The inclusion of the prior allows the method to borrow information from other nodes to estimate the embedding vector for a particular node. The experiment section also shows the importance of norm penalization. Regarding the choice of ℓ_2 norm over ℓ_1 norm, we tried both norms in the process of coming up with R-NS. In many cases ℓ_2 performed better and we did not have a motivation for the sparsity of the embedding vectors. Therefore we used ℓ_2 norm penalization.

Table 1: Data sets for experiments

Data Set	$\|V\|$	$\|E\|$	# Labels
BlogCatalog	10312	333983	39
Flickr	7564	239365	9
Wikipedia	4777	184812	40

Scalability The R-NS objective has a different negative sampling distribution for different nodes, so we need a method to draw negative samples from the desired distribution without changing the time complexity. We suggest two methods for adaptive sampling, depending on whether the network is densely connected or not. The details of the two suggested methods are presented in the supplementary material.

Experiments

In this section we show the superiority of R-NS to NS for multi-label classification tasks on several large-scale networks. We also illustrate in the appendix the undesirable effects of the popular neighbor problem on the embedding vectors through the analysis of a small real network and we show how R-NS alleviates the problem.

As we discussed earlier there are many embedding algorithms which use NS to approximate the SGA objective. Therefore to compare R-NS, NS, and several other possible alternative for NS, we need to pick one of those embedding algorithms and make a comparison among R-NS and its alternatives. We chose LINE because it outperforms methods like Node2vec and DeepWalk on some network datasets, according to the evaluation results of (Wang et al. 2017). Moreover, LINE is specifically designed for network data, while many of the other methods change the network information to a sequence representation and use natural language processing methods. As a side note, the way LINE defines neighborhoods (the set of immediate neighbors of the the nodes in the network) allows it to emulate other embedding algorithms like Struc2vec, Node2vec, DeepWalk. The reason is provided in the supplementary file.

Table 2: Algorithm comparison for 50% data labeled, Type I and II objectives. The maximum standard deviation of the presented numbers for type I and type II are respectively 0.005 and 0.0072. The exact standard deviation of the methods and ROC curves can be found in the supplementary file.

Type I / Type II	Micro-F_1			Macro-F_1		
Algorithm	Blog Catalog	Flickr	Wikipedia	Blog Catalog	Flickr	Wikipedia
NS	0.36 / 0.12	0.54 / 0.46	0.47 / 0.49	0.25 / 0.07	0.43 / 0.35	0.15 / 0.15
NN-NS	0.32 / 0.15	0.58 / 0.41	0.46 / 0.45	0.22 / 0.09	0.46 / 0.31	0.14 / 0.11
R-NS ($\lambda = 0$)	0.34 / 0.15	0.55 / 0.40	0.42 / 0.44	0.23 / 0.09	0.43 / 0.28	0.11 / 0.11
R-NS	**0.38 / 0.25**	**0.60 / 0.57**	**0.48 / 0.54**	**0.27 / 0.17**	**0.49 / 0.47**	**0.15 / 0.20**
R-NS λ	0.046 / 0.059	0.043 / 0.056	0.053 / 0.067	0.046 / 0.059	0.043 / 0.056	0.053 / 0.067

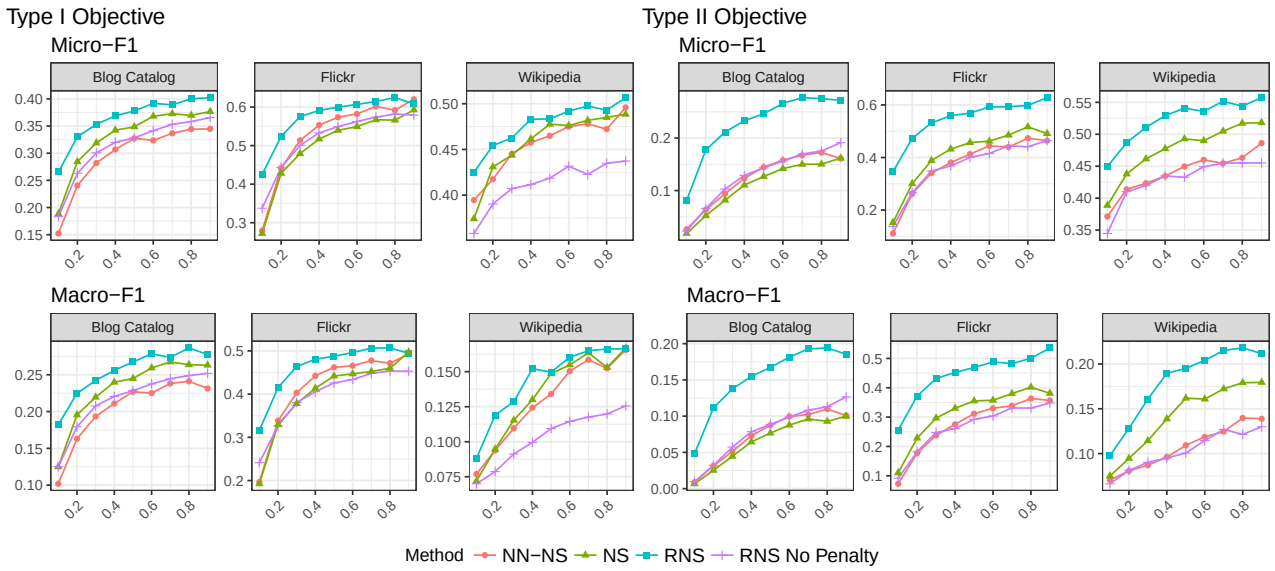

Figure 1: Multi-label classification performance comparison on the datasets with varying amounts of labeled training data, averaged over 5 iterations. The x axis is the fraction of data with labels, the y axis is the F_1 score. For both types of objective RNS leads to improved performance.

Multi-label classification

The multi-label classification task entails predicting the possibly multiple labels associated with each node in the given network. A fraction of nodes and their labels are observed. This training set information is used to predict the labels of the nodes in the test set, the remaining nodes. This is a common task for evaluating network embeddings (Huang et al. 2018).

Data Sets We use the following datasets. The BlogCatalog dataset (Zafarani and Liu 2009) is a friendship network of bloggers on the BlogCatalog website. The labels are categories which the bloggers use to tag their posts. The Flickr dataset (Huang, Li, and Hu 2017a) is a network of interactions between Flickr users. The labels represent the interest groups of the users such as "black and white photos". The Wikipedia dataset (Mahoney 2011) is the word co-occurrence network of the Wikipedia dataset, which used a 2-word window to determine co-occurrence edges. They used the Stanford POS-Tagger to infer parts of speech for words in the network, and treat these as the labels.

Compared Algorithms. To validate the performance of our approach we compare R-NS with NS and several alternatives to R-NS that also can scale up to large networks.

- Non-neighbor Negative Sampling (NN-NS). This algorithm differs from the R-NS procedure explained in the previous section in the negative sampling distribution P_u, where a node also can chose itself as a negative sample. To be more specific, $P_u^{\text{self sample}}(v) \propto d_v^\beta$ for $v \in V - N(u)$ but the probability of choosing $v \in N(u)$ is zero. One can show that self negative sampling in the type I objective case is equivalent to having a penalty on the norm.
- R-NS (no penalty). This algorithm is different form R-NS in that it does not penalize the norms of the embedding vectors which is equivalent to $\lambda = 0$.
- R-NS. This is the R-NS algorithm introduced in the pre-

vious section which optimizes the objective that is the summation of terms like (6) for all pairs of a node and its neighbor.

Given the empirical results in both natural language processing and network embedding demonstrating the superiority of NS versus competing methods (Mikolov et al. 2013b), we did not include direct optimization of SGA objective, noise contrastive estimation, nor hierarchical softmax in the baseline methods.

Parameter settings mostly follow previous literature. The penalty coefficient λ is calculated based on the formula given in the paper. The degree power β is chosen to be $3/4$, which is a widespread default in the literature. Exact settings are presented in the supplementary file.

Experimental Results The network embeddings are used in a one-vs-all logistic regression with L2 penalty, implemented through the LibLinear library. From Table 2, we see that simply excluding neighbors from negative sampling is insufficient, as R-NS(no penalty) underperforms even the standard negative sampling scheme. This also demonstrates the importance of the norm penalization. NN-NS is presented to show the importance of deriving a formula for the penalty coefficient. NN-NS can be perceived as R-NS with a penalty coefficient on the norm where the penalty depends on the degree of the node. This is the case because NN-NS, like R-NS, does not let negative samples be from the neighbors. And since it lets a node choose itself as a negative sample with probability related to the degree, it can be considered to do degree-related norm penalization. NN-NS does not have enough freedom on the form of the penalty, unlike R-NS, and that can be seen as the main reason that it does not perform as well as R-NS. See Table 2 for a comparison of the algorithms.

To provide further intuition about the performance of the methods, we conduct experiments where we randomly divide the data into train and test splits and increase the proportion of training examples from 10% to 90% of the data in increments of 10%. The results for each dataset, averaged over multiple iterations, are summarized in Figure 1. We see that R-NS leads to a significant improvement over NS in both multi-label Micro-F_1 and Macro-F_1 scores for embeddings using both the Type I and Type II objectives for most train-test splits. It also improves on NN-NS and R-NS (no penalty). For most of the experimental set ups NN-NS and R-NS with no penalty do not provide much improvement over NS and in fact underperform NS for Type I and Type II embeddings on the Blog Catalog and Wikipedia datasets. In particular for the Blog Catalog network with Type II embeddings, R-NS outperforms NS in terms of Micro-F_1 score by about 94% for 50% of the training data labeled. Meanwhile for the Flickr data with Type II embeddings R-NS outperforms NS in terms of Micro-F_1 score by about 126% for only 10% of the training data labeled.

To see the importance of choosing a proper λ, and to gain a general intuition for how λ effects the quality of the embeddings, we also vary the λ in R-NS and compare the result with NS. We clarify again that the result of the experiments in Table 2 and Figure 1 are based on the formula which is given in the paper, not by searching over the grids of the λ.

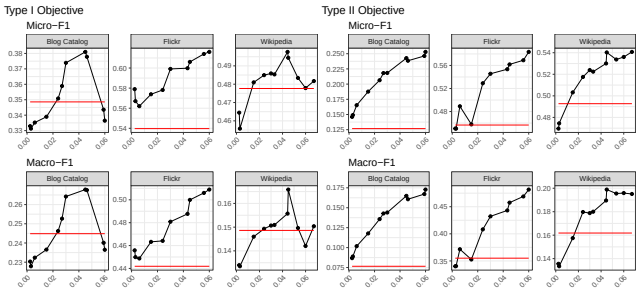

Figure 2: Effect of penalty λ on multi-label classification performance for data with 50% of labels withheld, averaged over 5 iterations. The red Horizontal line indicates performance of NS.

Figure 2 shows the effect of different λ settings on F_1 scores when the train-test split is set to 50%, averaged over 5 iterations. Note that especially in the case of the Blog Catalog and Wikipedia Network datasets λ should be within a suitable region in order for R-NS to outperform NS. Since in all of the experimental results R-NS is superior to NS when it uses the presented formulation for λ, the formula assigns a suitable value to λ based on the network.

Source code for running experiments and data are available online at https://github.com/delimited0/network_embedding.

Conclusion

In this paper we examined the performance of the NS approximation to the SGA objective for network embedding. We showed NS suffers from the Popular Neighbor Problem, which can lead to embedding vectors with undesirable properties that are contrary to the goals of the SGA objective and network embedding.We proposed R-NS, which avoids the Popular Neighbor Problem. We demonstrated the efficiency and effectiveness of R-NS by doing experiments on several real-world networks. Moreover, because of the careful design of R-NS, the method is scalable and can be employed for the learning of embeddings in an unsupervised fashion.

For future work we plan to apply R-NS to other areas where NS is widely used, such as natural language processing, and evaluate its performance. We also plan to investigate more intelligent schemes for choosing negative samples that use more information than just the degree of a node and the set of its neighbors. For example we could consider the structure of the nodes and its measure of connectivity when taking negative samples.

Acknowledgments

We thank the anonymous reviewers for their feedback. This work is, in part, supported by NSF (#IIS-1750074 and #IIS-1718840).

References

Bhagat, S.; Cormode, G.; and Muthukrishnan, S. 2011. *Node Classification in Social Networks*. Springer. 115–148.

Cao, S.; Lu, W.; and Xu, Q. 2016. Deep neural networks for learning graph representations. In *AAAI*, 1145–1152.

Chen, L.; Yuan, F.; Jose, J. M.; and Zhang, W. 2018. Improving negative sampling for word representation using self-embedded features. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 99–107. ACM.

Cui, P.; Wang, X.; Pei, J.; and Zhu, W. 2018. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2001. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:.

Grover, A., and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16.

Gutmann, M. U., and Hyvärinen, A. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research* 13(Feb):307–361.

Huang, X.; Song, Q.; Li, J.; and Hu, X. 2018. Exploring expert cognition for attributed network embedding. In *ACM International Conference on Web Search and Data Mining*, 270–278.

Huang, X.; Li, J.; and Hu, X. 2017a. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, 633–641. SIAM.

Huang, X.; Li, J.; and Hu, X. 2017b. Label informed attributed network embedding. In *ACM International Conference on Web Search and Data Mining*, 731–739.

Lai, Y.; Hsu, C.; Chen, W.; Yeh, M.; and Lin, S. 2017. PRUNE: preserving proximity and global ranking for network embedding. In *NIPS*, 5263–5272.

Liben-Nowell, D., and Kleinberg, J. 2003. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Mahoney, M. 2011. Large text compression benchmark.

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 3111–3119. USA: Curran Associates Inc.

Mnih, A., and Teh, Y. W. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Morin, F., and Bengio, Y. 2005. Hierarchical probabilistic neural network language model. volume 5, 246–252. Citeseer.

Mu, C.; Yang, G.; and Yan, Z. 2018. Revisiting skip-gram negative sampling model with regularization. *arXiv preprint arXiv:1804.00306*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, 701–710.

Ribeiro, L. F.; Saverese, P. H.; and Figueiredo, D. R. 2017. Struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17.

Sadeghian, A.; Rodriguez, M.; Wang, D. Z.; and Colas, A. 2016. Temporal reasoning over event knowledge graphs.

Shaw, B., and Jebara, T. 2009. Structure preserving embedding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 937–944. New York, NY, USA: ACM.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, 1067–1077.

Tu, C.; Zhang, W.; Liu, Z.; Sun, M.; et al. 2016. Max-margin deepwalk: Discriminative learning of network representation. In *IJCAI*, 3889–3895.

Tu, K.; Cui, P.; Wang, X.; Yu, P. S.; and Zhu, W. 2018. Deep recursive network embedding with regular equivalence. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2357–2366. ACM.

Wang, X.; Cui, P.; Wang, J.; Pei, J.; Zhu, W.; and Yang, S. 2017. Community preserving network embedding. In *AAAI*, 203–209.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 40–48. JMLR.org.

Zachary, W. 1977. An information flow model for conflict and fission in small groups. *J. of Anthropological Research* 33:452–473.

Zafarani, R., and Liu, H. 2009. Social computing data repository at ASU.

Zhang, Z.; Cui, P.; Wang, X.; Pei, J.; Yao, X.; and Zhu, W. 2018. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2778–2786. ACM.