A Faster Converging Negative Sampling for the Graph Embedding Process in Community Detection and Link Prediction Tasks

Kostas Loumponias[a], Andreas Kosmatopoulos[b], Theodora Tsikrika[c], Stefanos Vrochidis[d] and Ioannis Kompatsiaris[e]

Information Technologies Institute, Centre for Research and Technology Hellas - CERTH, GR-54124, Thessaloniki, Greece

Keywords: Skipgram Algorithm, Negative Sampling, Graph Embedding, Community Detection, Link Prediction.

Abstract: The graph embedding process aims to transform nodes and edges into a low dimensional vector space, while preserving the graph structure and topological properties. Random walk based methods are used to capture structural relationships between nodes, by performing truncated random walks. Afterwards, the SkipGram model with the negative sampling approach, is used to calculate the embedded nodes. In this paper, the proposed SkipGram model converges in fewer iterations than the standard one. Furthermore, the community detection and link prediction task is enhanced by the proposed method.

1 INTRODUCTION

Neural network applications have expanded significantly in recent years in various scientific fields, such as image classification (He et al., 2019), natural language processing (Chowdhary, 2020), and network analysis (Nguyen et al., 2018). A particularly successful utility of deep learning is the embedding process (Cai et al., 2018), which is used for mapping discrete variables to continuous vectors. This technique has found practical applications in graph embedding, enabling the use of such methods for network analysis tasks, such as community detection (Rozemberczki et al., 2019; Cavallari et al., 2017) and link prediction (Grover and Leskovec, 2016).

Graph embedding is an approach that is utilised to transform nodes, edges, and their features into a lower dimensional vector space, while the properties, like the graph structure and topological information, are preserved. Graph embedding is a complex process, since the graphs may vary in terms of their scale and specificity. Therefore, a variety of approaches for embedded graphs have been proposed (Goyal and Ferrara, 2018), each with a different level of granularity. In particular, the embedding methods can be categorised as follows: (a) factorization based, (b) deep learning based, and (c) random walk based.

Factorization based algorithms aim to factorize the matrix that represents the connections between the nodes of the graphs, in order to obtain the embedding. In the case where the obtained matrix is positive semidefinite, such as the Laplacian matrix, the eigenvalue decomposition can be utilized. Deep learning-based algorithms attempt to preserve the first and second order network proximities by using deep autoencoders, due to their ability to model non-linear structure in the data. Random walk based algorithms are used to capture structural relationships between nodes, by performing truncated random walks. Therefore, a graph is transformed into a collection of node sequences, in which, the occurrence frequency of a vertex-context pair measures the structural distance between them (Zhang et al., 2018). Some of the most popular random walk based methods are DeepWalk (DW) (Perozzi et al., 2014) and node2vec (n2v) (Grover and Leskovec, 2016).

DW aims to learn embedded nodes via a random walk sampling process and the word2vec algorithm (Mikolov et al., 2013a). In word2vec, the SkipGram model is applied by using sentences (series of words) to train the embedded word. In a nutshell, the training objective is to use the center word in order to predict surrounding words of the sentence. Hence, the DW process initially starts from an arbitrary node and performs transitions to neighboring nodes in an uniform fashion (i.e. all neighbors of a node have equal

[a] https://orcid.org/0000-0002-6268-3893
[b] https://orcid.org/0000-0001-5334-741X
[c] https://orcid.org/0000-0003-4148-9028
[d] https://orcid.org/0000-0002-2505-9178
[e] https://orcid.org/0000-0001-6447-9020

Loumponias, K., Kosmatopoulos, A., Tsikrika, T., Vrochidis, S. and Kompatsiaris, I.
A Faster Converging Negative Sampling for the Graph Embedding Process in Community Detection and Link Prediction Tasks.
DOI: 10.5220/0011142000003277
In *Proceedings of the 3rd International Conference on Deep Learning Theory and Applications (DeLTA 2022)*, pages 86-93
ISBN: 978-989-758-584-5; ISSN: 2184-9277

probability of being selected). A walk process is terminated upon reaching a maximum length which is predefined (user-provided parameter). This process is applied for all nodes and is repeated n times (number of walks). It follows that a node and a series of nodes can be treated as a word and a sentence, respectively, enabling, the SkipGram algorithm to be naturally utilized for generating node embeddings.

One of the disadvantages of DW is that it cannot control the path generated by the random walk. In order to address this limitation, the n2v approach has been proposed. In particular, n2v uses random walks (similarly to DW) with transition probabilities that are governed by weights, i.e., n2v generates biased walks. Instead of performing walking steps randomly, the concepts of Breadth-First-Search (BFS) and Depth-First-Search (DFS) sampling are introduced to control random behavior.

An important characteristic of the original SkipGram implementation is that it's based in the use of the softmax function which can result in a very expensive computational cost. To alleviate this problem the negative sampling process is utilised which operates under a lower computational complexity cost and typically offers better execution times. The negative sampling process uses the sigmoid function to differentiate the actual context nodes (positive) from randomly drawn nodes (negative). The negative samples are selected via the noise distribution (Mikolov et al., 2013b). However, the main drawback of the sigmoid function is the vanishing gradient problem (Hochreiter, 1998) which ultimately leads to a slower convergence of the SkipGram model.

In this paper, a modified SkipGram algorithm is proposed for the DW and n2v processes to achieve faster convergence in community detection and link prediction tasks compared to the standard DW and n2v processes, preserving the accuracy in both tasks.More specifically, a novel function, $\sigma_b(x)$, with an additional trainable parameter is proposed to tackle the limitations of a standard sigmoid function. In addition, the new gradients in the backward propagation process are calculated and explained in detail. Finally, the calculated embedded nodes are used in the community and link prediction task. More specifically, and after producing the graph embeddings, the k-means algorithm (Hartigan and Wong, 1979) is executed to obtain the final communities, while the logistic regression model (Hosmer Jr et al., 2013) with the consideration of various similarity measures is utilised to predict the existence of links between two nodes in the graph. The experimental results in real graphs show that the proposed method converges (i.e. the k-means algorithm and logistic regression model

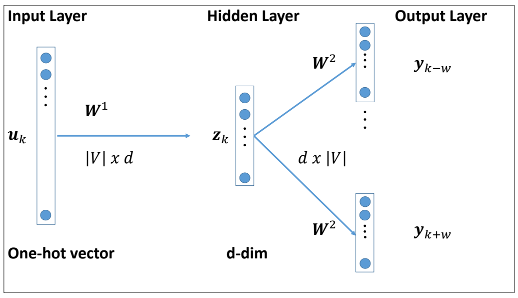

Figure 1: The SkipGram model.

achieve their optimal results) faster and provides more coherent community detection and link predictions results than the standard DW and n2v processes.

2 PROPOSED METHOD

A general overview and mathematical formulation of the SkipGram algorithm under the negative sampling approach is provided. Following that, the proposed DW (and n2v) algorithm using a modified SkipGram model is described.

2.1 SkipGram Model: The Negative Sampling Approach

Let $G = (V, E)$ be a graph, where V and $E \subseteq (V \times V)$ correspond to the node and edge set of the graph, respectively. Figure 1 illustrates the SkipGram model where it can be seen that it corresponds to a fully connected neural network with one hidden layer and multiple outputs. The SkipGram model attempts to predict the context nodes of a *sentence* (a series of nodes) given the target node. The input vector $\mathbf{u}_k$ is the one-hot embedded vector of target node $u_k \in V$, $\mathbf{W}^1 \in M_{|V|\times d}$ is the embedding matrix, where $|V|$ denotes the total number of the nodes and d the embedding size. Each row of $\mathbf{W}^1$ represents the embedded vector of node $u_i \in V$. The vector $\mathbf{z}_k \in \mathbb{R}^d$ stands for the embedded vector of node u_k and is equal with

$$\mathbf{z}_k = \mathbf{W}^{1'} \cdot \mathbf{u}_k. \tag{1}$$

$\mathbf{W}^2 \in M_{d\times|V|}$ is the output embedding matrix, while $\{\mathbf{y}_{k-w}, \ldots, \mathbf{y}_{k-1}, \mathbf{y}_{k+1}, \ldots, \mathbf{y}_{k+w}\}$ are the predicted context nodes (one-hot vectors) when the input-target node is u_k, where w denotes the window size.

In the vanilla SkipGram model, the cost function J is calculated via the softmax function as

$$J(\theta) = -\sum_{c=1}^{C} \log \frac{\exp\left(\mathbf{W}_c^{2'} \cdot \mathbf{z}_k\right)}{\sum_{i=1}^{|V|} \exp\left(\mathbf{W}_i^{2'} \cdot \mathbf{z}_k\right)}, \tag{2}$$

where $\theta = [\mathbf{W}^1, \mathbf{W}^2]$ and $\mathbf{W}_i^2$ stands for the i-th column of the output embedding matrix $\mathbf{W}^2$ and C denotes the total number of context nodes. Furthermore, the term

$$P(u_c|u_k;\theta) = \frac{\exp\left(\mathbf{W}_c^{2'} \cdot \mathbf{z}_k\right)}{\sum_{i=1}^{|V|} \exp\left(\mathbf{W}_i^{2'} \cdot \mathbf{z}_k\right)} \tag{3}$$

represents the conditional probability of observing a context node u_c given the target node u_k.

From (2), it follows that the softmax function is computationally expensive, as it requires scanning through the entire output embedding matrix $\mathbf{W}^2$ to compute the probability distribution of all nodes $\in V$. Furthermore, the normalization factor in the denominator (2) also requires $|V|$ iterations. Due to this computational inefficiency, softmax function is not utilised in most implementations of SkipGram.

Thereafter, the negative sampling process with sigmoid function is used, which reduces the complexity of the algorithm. More specifically, for each positive pair, $\{u_k \text{ and } u_c\}$ in the training sample, K number of negative samples are drawn from the noise distribution $P_n(w)$ (Mikolov et al., 2013b), and the model will update $(K+1) \times d$ neurons in the matrix $\mathbf{W}^2$, where K is usually set equal to 5. Thus, the logarithm of the conditional probability (3) is approximated by

$$\log P(u_c|u_k;\theta) = \log\sigma\left(\mathbf{W}_c^{2'} \cdot \mathbf{z}_k\right) + \sum_{i=1}^{K} \log\sigma\left(-\mathbf{W}_{neg(i)}^{2'} \cdot \mathbf{z}_k\right), \tag{4}$$

where $\sigma(x)$ is the sigmoid function, while $\{\mathbf{W}_{neg(i)}^2\}_{i=1}^K$ is the set of columns $\mathbf{W}_i^2$ (vectors $d \times 1$) which are randomly selected from the noise distribution $P_n(w)$. The first term of (4) indicates the logarithmic probability of the positive sample u_c to appear within the context window of the target node u_k, while, the second term indicates the sum of the logarithmic probabilities of the negative samples $u_{neg(i)}$ not appearing in the context window.

In the negative sampling process, only $K+1$ columns of the output embedding matrix $\mathbf{W}^2$ are updated, while in the embedding matrix $\mathbf{W}^1$ only one row is updated (let $\mathbf{W}_s^1 \in \mathbb{R}^d$), since the input $\mathbf{u}_k$ is one-hot vector. Then, the update equations, through the backward propagation, are

$$\mathbf{c}_j = \mathbf{c}_j - \eta \cdot (\sigma(x_j) - t_j) \cdot \mathbf{z}_k, \tag{5}$$

$$\mathbf{W}_s^1 = \mathbf{W}_s^1 - \eta \cdot \sum_{j=1}^{K+1} (\sigma(x_j) - t_j) \cdot \mathbf{c}_j, \tag{6}$$

where

$$\mathbf{c}_j = \begin{cases} \mathbf{W}_c^2, & j = 1 \\ \mathbf{W}_{neg(j-1)}^2, & j = 2, \ldots, K+1 \end{cases},$$

$$t_j = \begin{cases} 1, & j = 1 \\ 0, & j = 2, \ldots, K+1 \end{cases},$$

$x_j = \mathbf{c}_j^{'} \cdot \mathbf{z}_k$ and η is the learning rate.

In the case where $j = 1$ (positive sample) and $x_j \to -\infty$, the term $-(\sigma(x_j) - t_j)$ is maximized. Therefore, the SkipGram model updates-corrects the weights $\theta = [\mathbf{W}^1, \mathbf{W}^2]$ for low values of x_j, otherwise, when the values of x_j are high, the updates, $-(\sigma(x_j) - t_j)$ are negligible. In the same way, for $j \neq 1$ (negative samples), it turns out that for low values of x_j, the updates are negligible. Thus, the inner product $x_j = \mathbf{c}_j^{'} \cdot \mathbf{z}_k$ defines a proximity between the nodes u_k and u_c.

2.2 SkipGram with a Modified Negative Sampling Process

The main drawbacks of the sigmoid function are the vanishing gradient of $\log\sigma(x)$ ($\nabla \log\sigma(x)$) and its saturated values. More specifically, as it can be seen in Figure 2, the gradient $\nabla \log\sigma(x)$ (blue curve) converges to 0 for $x \to +\infty$ and to 1 for $x \to -\infty$. Thus, in the case where $j = 1$ (positive sample) for every low value of x_j the gradient is approximately equal to 1. Hence, the range of the updates is limited and the highest value that it can reach is equal to 1. This can cause a lot of computational burden until the weights θ converge.

In order to overcome the above limitation, the sigmoid b function is proposed

$$\sigma_b(x) = \frac{1}{1 + \exp(-b \cdot x)}, \tag{7}$$

where $b > 1$. It is worth noting that the range of the proposed function is the interval $[0,1]$, since it approximates the probability (4). In Figure 2, the derivative of $\log\sigma_b(x)$ (9) for $b = 2$ is illustrated. As it can be seen the range of the proposed derivatives is $[0,2]$.

The logarithm of the conditional probability (3) in the proposed method is equal to

$$\log P(u_c|u_k;\theta) = \log\sigma_b\left(\mathbf{W}_c^{2'} \cdot \mathbf{z}_k\right) + \sum_{i=1}^{K} \log\sigma_b\left(-\mathbf{W}_{neg(i)}^{2'} \cdot \mathbf{z}_k\right). \tag{8}$$

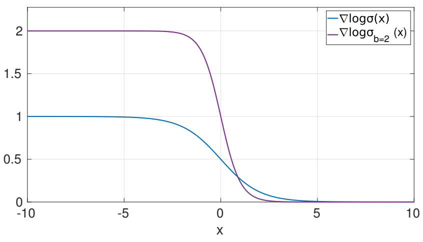

Figure 2: The derivatives of $\log\sigma(x)$ and $\log\sigma_{b=2}(x)$.

Then, it is proved that the derivative of $\log\sigma_b(x)$ is equal with

$$\nabla \log\sigma_b(x) = \frac{\partial\sigma_b(x)}{\partial x} = b\cdot(1-\sigma_b(x)). \quad (9)$$

Thus, using (8) and (9) the update equations for the proposed SkipGram model (SkipGram$_b$) are defined as

$$\mathbf{c}_j = \mathbf{c}_j - \eta\cdot b\cdot(\sigma_b(x_j)-t_j)\cdot\mathbf{z}_k, \quad (10)$$

$$\mathbf{W}_s^1 = \mathbf{W}_s^1 - \eta\cdot b\cdot\sum_{j=1}^{K+1}(\sigma_b(x_j)-t_j)\cdot\mathbf{c}_j. \quad (11)$$

It is important to note that, in the proposed process, the additional parameter b is trainable and the update equation (using (8) and (9)) is proved to be equal with

$$b = b - \eta_b\cdot\sum_{j=1}^{K+1}(\sigma_b(x_j)-t_j)\cdot x_j, \quad (12)$$

where η_b is the learning rate for the parameter b. From (12), it can be straightforwardly deduced that the updates of parameter b will be negligible when the error $(\sigma_b(x_j)-t_j)$ is reduced. It is derived from the above results that the proposed sigma function can be applied to any machine learning model instead of the standard sigma function, however the updates equations (10)-(12) need to customized to the applicable machine learning model.

It is worth mentioning that in literature there have been alternative activation functions proposed aiming for better performance in various scientific fields. In (Banerjee. et al., 2021), the SM-Taylor softmax function has been applied for image classification tasks and the results showed that it outperforms the normal softmax function.

Next, the proposed process (Algorithm 1) for community detection is presented. The proposed method, DeepWalk$_b$ (DW$_b$) has similar framework to DW, with the main difference being the use of the SkipGram$_b$ model instead of the standard one. In the following, lines $1-7$ in (Algorithm 1) represent the DW$_b$ process. Then, the $k-$means algorithm is used to detect the communities of the graph (*Com*). As a final remark, we note that the proposed SkipGram$_b$ model can be also applied to the n2v process, since the only difference between DW and n2v is the way that random walks are performed (line 4 of Algorithm 1).

In Algorithm 2 the process of link prediction as well as the evaluation process is presented. More specifically, three sub-graphs G_{tr}, G_{mod} and G_{ts} are derived from the initial graph G. Initially, the *train* graph G_{tr} is used to calculate the embedded nodes θ_{tr}. Then, the similarities between embedded nodes are calculated using various operators (*oprtr*), such as Hadamard product, $L1$, $L2$ norm (Luo et al., 2016) etc., afterwards the logistic regression model is used to calculate the classifiers (whether the nodes are connected or not). Next, the classifiers (one for each operator) are evaluated considering the *model selection* graph G_{mod} and the embeddings θ_{tr}. Finally, the operator with the highest accuracy score is utilised to evaluate the classifier for the *test* graph G_{ts}.

Algorithm 1: DeepWalk$_b$ for community detection.

```
Require: Graph $G$, number of communities $k$, window size $w$, embedding size $d$, walk length $t$, number of walks $n$
1: for i=1:n do
2:     $O = Shuffle(V)$
3:     for $u_i \in O$ do
4:         $RW_{u_i}$ = RandomWalk($G, u_i, t$)
5:         $\theta$ = SkipGram$_b$($RW_{u_i}, w$)
6:     end for
7: end for
8: $Com$ = $k$-means($\theta$,$k$)
9: return $Com$
```

Algorithm 2: DeepWalk$_b$ for link prediction.

```
Require: Graph $G$, window size $w$, embedding size $d$, walk length $t$, number of walks $n$
1: $G_{tr}$, $G_{mod}$, $G_{ts}$ = split($G$)
2: $\theta_{tr}$ = DW$_b$($G_{tr}$, $w$, $d$, $t$, $n$)
3: for i=1:$\nu_{oprtr}$ do
4:     clsfr(i) = Logistic Regression($\theta_{tr}$, operator(i))
5:     AccScore(i) = evaluate(clsfr(i), $G_{mod}$, $\theta_{tr}$, oprtr(i))
6: end for
7: $i_{max}$ = argmax(AccScore)
8: $clsfr_{max}$, $oprtr_{max}$ = clsfr($i_{max}$), oprtr($i_{max}$)
9: $\theta_{ts}$ = DW$_b$($G_{tsr}$, $w$, $d$, $t$, $n$)
10: Test Score = evaluate($clsfr_{max}$, $G_{ts}$, $\theta_{ts}$, $oprtr_{max}$)
```

3 EXPERIMENTAL EVALUATION

In this section, we conduct experimental evaluation on the proposed methods, DW$_b$ and n2v$_b$ (n2v using SkipGram$_b$ algorithm) against the standard DW and n2v process, respectively. To that end, real datasets with ground-truth communities are used. We use publicly available graphs Cora, Pubmed and CiteSeer (Sen et al., 2008) for evaluation, which are provided with ground-truth communities. The Cora dataset consists of 2708 publications, classified into 7 classes, and 5429 links. The PubMed dataset consists of 19717 publications, classified into 3 classes, and 44338 links. The CiteSeer consists of 3312 publications, classified into 6 classes, and 4732 links.

The parameters used in methods DW$_b$, n2v$_b$, DW and n2v are $w = 10$ (window size), $d = 128$ (embedding size), $t = 80$ (walk-length) whereas the number of epochs and the batch-size are equal to 10 and 1000, respectively. Parameters p and q used in n2v$_b$ and n2v were evaluated using a grid search over [0.25, 0.5, 1, 2, 4] (Grover and Leskovec, 2016). The experimental sets are conducted considering different values of learning rate, $\eta \in \{0.05, 0.2, 0.6, 1.0, 2.0\}$. Furthermore, in the proposed processes, the learning rate of the parameter b is set equal to $\eta_b = 0.01$ for all experimental sets. This value is often considered as a typical value for the learning rate, while our experiments with η_b values close to it provided a stable training in the experimental sets; in future work, further values of η_b will also be considered. Finally, all methods use the stochastic gradient descent technique with momentum 0.9 for the back propagation process.

Regarding the community detection task, in each experiment, the Adjusted Rand Index (ARI), the Normalized Mutual Information (NMI) and the graph's modularity (Mod) (Vinh et al., 2010) are calculated, while, for the link prediction task the Area Under the Curve (AUC) score (Fawcett, 2006) is calculated. Furthermore, the StellarGraph tool (Grover and Leskovec, 2016) is used to split the input graph $G = (V,E)$ to $G_{tr} = (V,E_{tr})$, $G_{mod} = (V,E_{mod})$ and $G_{ts} = (V,E_{ts})$. In all datasets, E_{ts} includes 90% of the total edges (E), while the E_{tr} and E_{mod} include 75% and 25% of E_{ts}, respectively.

3.1 Evaluation of the Modified DeepWalk and Node2vec

In this section, the performance of the DW, DW$_b$, n2v and n2v$_b$ methods is presented. Figure 4 illustrates the performance of DW and DW$_b$ considering the metrics ARI, NMI, Mod and AUC (y axis) for different values of η in the CiteSeer, Cora and PubMed graph. Moreover, the x axis in the sub-figures stands for the number of epochs. It is clear, that the proposed method, DW$_b$, converges faster than the standard one, DW, in all datasets for all the different values of η. Additionally, it can be observed that the DW process for $\eta = 0.05$, does not converge within 10 epochs (i.e more than 10 epochs are required) in any of the datasets.

In case of a lower value of η, the convergence speed will be further reduced, therefore, lower learning rates are not considered in this paper. As an example, a learning rate equal to 0.005 is used for the CiteSeer graph to verify the above statement. Figure 3 illustrates the loss values of DW and DW$_b$ in the community detection task for each epoch. It is clear that the DW process is not able to converge within 100 epochs, while DW$_b$ can converge to 30 epochs, due to the trainable parameter b.

More details about the convergence speed in community detection and link prediction task for DW and DW$_b$ are provided in Table 1. The columns *cd epochs* and *lp epochs* stand for the required number of epochs in order for the community detection metrics (i.e. ARI, NMI, Mod) and link prediction metric (AUC) to converge. It is demonstrated that the DW$_b$ process outperforms the standard DW, considering the convergence speed (in both tasks) in all experimental sets apart from one. More specifically, only in the PubMed graph for $\eta = 2.0$, the two methods require the same number of epochs. Moreover, it is worth noting that DW$_b$ is more robust in terms of converging speed at different values of η than the standard DW, since DW$_b$ uses the introduced trainable parameter b.

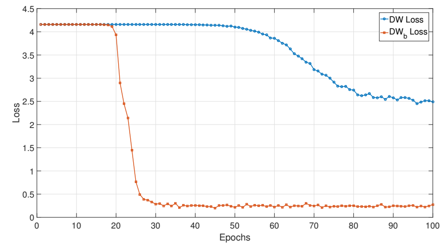

Figure 3: The loss values of DW and DW$_b$ in community detection task, for the CiteSeer graph with $\eta = 0.005$.

In addition, the proposed method provides the same (or better) performance as DW in the community detection and link prediction tasks in fewer epochs. In Table 1, the best performances of all metrics, within 10 epochs, for both methods are provided. As it can be seen, the best performances of the metrics, regardless of the learning rate, are similar (the differences are less than 10^{-2}) for both methods in most experiments. However, in the CiteSeer graph, DW$_b$ has ARI score equal to 0.1409 ($\eta = 2.0$), while the highest ARI score for DW is equal to 0.1235 ($\eta = 2.0$). In addition, the highest Mod and AUC score for DW$_b$ are equal to 0.7358 ($\eta = 1.0$) and 0.9207 ($\eta = 0.6$), respectively, while the highest values of DW are equal to 0.7224 ($\eta = 2.0$) and 0.9105 ($\eta = 2.0$). Finally, in Pubmed graph, DW$_b$ has AUC score equal to 0.8674 ($\eta = 0.6$), while the highest AUC score for DW is equal to 0.7779 ($\eta = 2.0$).

The performances of n2v and n2v$_b$ in the community detection and link prediction tasks are provided in Table 2 and Figure 5, in a similar way as in Table 1 and Figure 4, respectively. It can be observed that the n2v$_b$ process outperforms the standard n2v process, considering the convergence speed (in both tasks), in all experimental sets, apart from the PubMed graph for $\eta = 2$, where both n2v and n2v$_b$ require 1 epoch to converge. As it can be seen in Table 2, n2v$_b$ provides a more robust converging speed (for both tasks) at different values of η than the standard n2v.

Table 1: The best performances of DW and DW$_b$ regarding the metrics ARI, NMI, Mod and AUC. The columns *cd epochs* and *lp epochs* contain the required number of epochs in order for the community detection metrics (i.e. ARI, NMI, Mod) and link prediction metric (AUC) to converge.

		ARI		NMI		Mod		AUC		cd epochs		lp epochs	
	DW \ DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$
CiteSeer	$\eta = 0.05$	0.003	0.1240	0.023	0.2565	0.1692	0.7193	0.4140	0.9090	>10	6	>10	5
	$\eta = 0.2$	0.1183	0.1276	0.2581	0.2628	0.7089	0.7266	0.6478	0.9137	7	2	>10	4
	$\eta = 0.6$	0.1198	0.1246	0.2572	0.2551	0.7159	0.7308	0.7005	**0.9207**	4	**1**	>10	**1**
	$\eta = 1.0$	0.1213	0.1323	0.2659	0.2604	0.7168	**0.7358**	0.9070	0.9148	2	**1**	5	**1**
	$\eta = 2.0$	0.1235	**0.1409**	0.2601	0.2623	0.7224	0.7233	0.9105	0.9172	**1**	**1**	3	**1**
	DW \ DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$
Cora	$\eta = 0.05$	0.0264	0.4019	0.0723	0.4751	0.1625	0.7449	0.6677	0.8856	>10	4	>10	5
	$\eta = 0.2$	0.4049	0.4001	0.4703	0.4747	0.7464	0.7426	0.8211	0.8975	7	2	>10	2
	$\eta = 0.6$	0.4044	0.3964	0.4761	0.4667	0.7437	0.7434	0.8857	0.8947	4	**1**	7	**1**
	$\eta = 1.0$	0.3721	0.3786	0.4560	0.4560	0.7461	0.7401	0.9009	0.9009	2	**1**	7	**1**
	$\eta = 2.0$	0.3834	0.3815	0.4510	0.4504	0.7390	0.7417	0.9010	0.8983	**1**	**1**	3	**1**
	DW \ DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$	DW	DW$_b$
PubMed	$\eta = 0.05$	-0.0001	0.3116	0.0003	0.2954	0.1200	0.6014	0.7114	0.7663	>10	3	>10	2
	$\eta = 0.2$	0.3110	0.3128	0.3027	0.3000	0.6017	0.5986	0.7722	0.7661	6	**1**	5	**1**
	$\eta = 0.6$	0.3129	0.3152	0.3012	0.2987	0.5988	0.6004	0.7714	**0.8674**	2	**1**	2	**1**
	$\eta = 1.0$	0.3132	0.3180	0.3008	0.2997	0.5997	0.5994	0.7717	0.7392	2	**1**	**1**	**1**
	$\eta = 2.0$	0.3140	0.3168	0.3003	0.2979	0.6013	0.6001	0.7779	0.8118	**1**	**1**	**1**	**1**

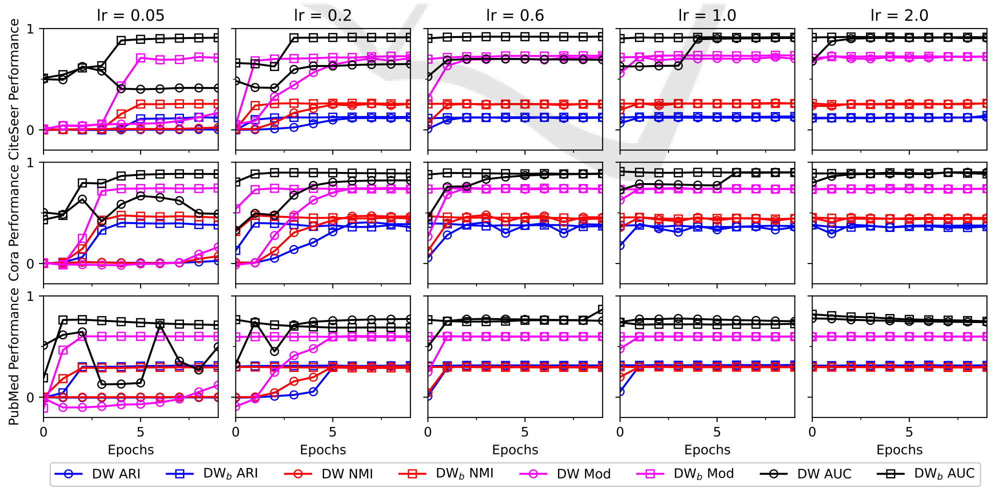

Figure 4: The performances of DW and DW$_b$ in community detection and link prediction task, considering the CiteSeer, Cora and PubMed datasets, for different values of learning rate.

As can be observed in Table 2, the best performances of the metrics, regardless of the learning rate, are similar for both methods in most experimental sets. However, in the CiteSeer graph, the highest ARI score for n2v$_b$ is equal to 0.1624 ($\eta = 0.6$), while the highest ARI score for n2v is equal to 0.1471 ($\eta = 2.0$). In the Cora graph, the highest AUC score for n2v$_b$ is equal to 0.9284 ($\eta = 2.0$), while the high-

Table 2: The best performances of n2v and n2v$_b$ regarding the metrics ARI, NMI, Mod and AUC. The columns *cd epochs* and *lp epochs* contain the required number of epochs in order for the community detection metrics (i.e. ARI, NMI, Mod) and link prediction metric (AUC) to converge.

		ARI		NMI		Mod		AUC		cd epochs		lp epochs	
	n2v \ n2v$_b$	n2v	n2v$_b$	n2v	n2v$_b$	n2v	n2v$_b$	n2v	n2v$_b$	n2v	n2v$_b$	n2v	n2v$_b$
CiteSeer	η = 0.05	0.003	0.1368	0.032	0.2634	0.1714	0.7483	0.4971	0.8861	>10	4	>10	4
	η = 0.2	0.1431	0.1567	0.2603	0.2709	0.7312	0.7334	0.7780	0.9079	7	2	>10	2
	η = 0.6	0.1341	**0.1624**	0.2561	0.2617	0.7330	0.7404	0.9166	0.9129	3	**1**	5	**1**
	η = 1.0	0.1471	0.1377	0.2631	0.2584	0.7376	0.74116	0.9230	0.9170	2	**1**	4	**1**
	η = 2.0	0.1430	0.1411	0.2674	0.2600	0.7309	0.7354	0.9258	0.9120	**1**	**1**	2	**1**
Cora	η = 0.05	0.0261	0.4114	0.0608	0.4789	0.1297	0.7475	0.6839	0.9147	>10	4	>10	6
	η = 0.2	0.4102	0.4085	0.4569	0.4772	0.7489	0.7446	0.7968	0.9266	7	2	>10	3
	η = 0.6	0.4018	0.4035	0.4669	0.4710	0.7440	0.7402	0.9041	0.9281	3	**1**	10	**1**
	η = 1.0	0.3838	0.4001	0.4621	0.4667	0.7448	0.7442	0.9119	0.9173	2	**1**	4	**1**
	η = 2.0	0.3917	0.4085	0.4594	0.4772	0.7469	0.7446	0.9125	**0.9284**	**1**	**1**	2	**1**
PubMed	η = 0.05	-0.0004	0.3185	0.0004	0.3016	0.1253	0.6014	0.8161	**0.8669**	>10	4	7	**1**
	η = 0.2	0.3087	0.3201	0.2962	0.2996	0.6004	0.6028	0.7669	0.7535	6	**1**	5	**1**
	η = 0.6	0.3139	0.3183	0.3014	0.2997	0.6013	0.5992	0.7803	0.7533	2	**1**	2	**1**
	η = 1.0	0.3162	0.3200	0.3032	0.2998	0.6023	0.6016	0.7853	0.7533	2	**1**	**1**	**1**
	η = 2.0	0.3162	0.3199	0.3034	0.2997	0.6013	0.6004	0.7856	0.8212	**1**	**1**	**1**	**1**

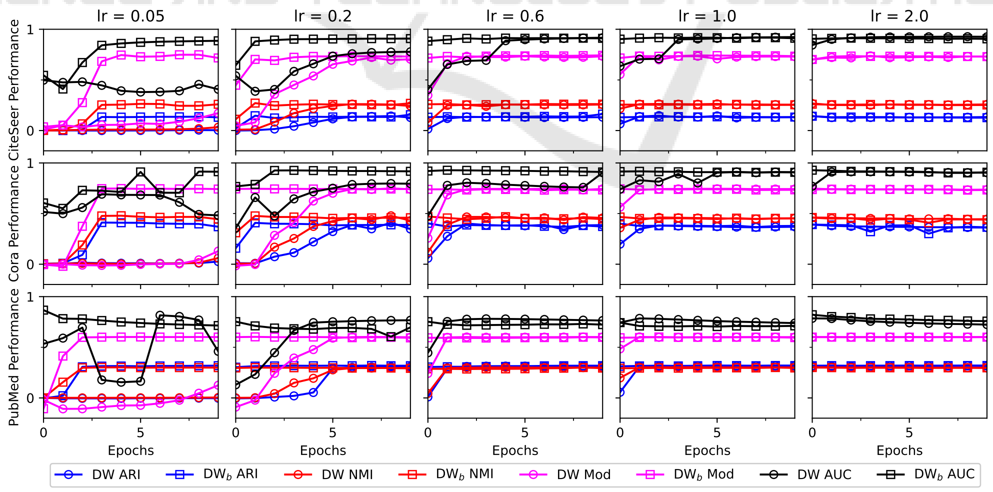

Figure 5: The performances of n2v and n2v$_b$ in community detection and link prediction task, considering the CiteSeer, Cora and PubMed datasets, for different values of learning rate.

est AUC score for n2v is equal to 0.9125 ($\eta = 2.0$). Finally, in the Pubmed graph, the highest AUC score for n2v$_b$ is equal to 0.8669 ($\eta = 0.05$), while for n2v is equal to 0.8161 ($\eta = 0.05$).

4 CONCLUSIONS

The aim of this paper was to improve the SkipGram model so that it converges faster. To that end, the sigmoid b function with the additional trainable parameter b was introduced to tackle the limitations of the standard sigmoid function used in the negative sampling process. The improved update equations of the SkipGram$_b$ model were used in the processes of DW and n2v for generating graph embeddings, while the k-means algorithm and the logistic regression model were utilised to detect the communities of the graph and to predict links between the nodes, respectively, using the (calculated) embedded nodes. The experimental results in real datasets showed that DW$_b$ and n2v$_b$ converged faster than DW and n2v, respectively, and attained higher ARI, Mod and AUC score in the CiteSeer graph, as well as higher AUC score in the Cora and PubMed graphs. Finally, the proposed SkipGram$_b$ provided a more robust performance in convergence speed than the standard SkipGram algorithm, considering different values of learning rates.

ACKNOWLEDGEMENTS

This research is part of projects that have received funding from the European Union's H2020 research and innovation programme under AIDA (GA No. 883596) and CREST (GA No. 833464).

REFERENCES

Banerjee., K., C.., V., Gupta., R., Vyas., K., H.., A., and Mishra., B. (2021). Exploring alternatives to softmax function. In *Proceedings of the 2nd International Conference on Deep Learning Theory and Applications - DeLTA,*, pages 81–86. INSTICC, SciTePress.

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

Cavallari, S., Zheng, V. W., Cai, H., Chang, K. C.-C., and Cambria, E. (2017). Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 377–386.

Chowdhary, K. (2020). Natural language processing. In *Fundamentals of Artificial Intelligence*, pages 603–649. Springer.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

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

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*, pages 855–864.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). *Applied logistic regression*, volume 398. John Wiley & Sons.

Luo, X., Chang, X., and Ban, X. (2016). Regression and classification using extreme learning machine based on l1-norm and l2-norm. *Neurocomputing*, 174:179–186.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., and Kim, S. (2018). Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976.

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*, pages 701–710.

Rozemberczki, B., Davies, R., Sarkar, R., and Sutton, C. (2019). Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining*, pages 65–72.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–106.

Vinh, N. X., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854.

Zhang, D., Yin, J., Zhu, X., and Zhang, C. (2018). Network representation learning: A survey. *IEEE transactions on Big Data*.