Asymmetric Node Similarity Embedding for Directed Graphs

Stefan Dernbach and Don Towsley

University of Massachusetts, Amherst
College of Information and Computer Sciences
{dernbach,towsley}@cs.umass.edu

Abstract. Node embedding is the process of mapping a set of vertices from a graph onto a vector space. Modern deep learning embedding methods use random walks on the graph to sample relationships between vertices. These methods rely on symmetric affinities between nodes and do not translate well to directed graphs. We propose a method to learn vector embeddings of nodes in a graph as well as the parameters of an asymmetric similarity function that can be used to retain the direction of relationships in the embedding space. The effectiveness of our approach is illustrated visually by the 2D embedding of a lattice graph as well quantitatively in multiple link prediction experiments on real world datasets.

Keywords: Node Embedding · Skip-Gram · Digraphs.

1 Introduction

Networks and graphs are ubiquitous in modern information settings. A graph's representational power of objects and relationships make them an essential tool in data visualization and processing. To further aid in many data processing tasks, we seek to learn a vector representation of the objects in a network. Many embedding schemes utilize the spectra of an affinity matrix of the graph to form vector representations of nodes [1, 2, 15]. These approaches, while effective, require eigen decompositions of large matrices and so do not scale well as the number of nodes in the graph increases. Recent embedding methods sample random walks from the graph and use a stochastic gradient descent process to learn vector representations for the nodes [10, 3]. These methods have been shown to be efficient on graphs scaling up to millions of nodes.

Implementing most undirected network embedding methods (such as those above) on a directed network requires making sacrifices to the network structure because these methods rely on symmetric affinities between nodes. This leads to the unrecoverable loss of the asymmetric relationships between nodes. In this work we propose a directed random walk based approach to learning node embeddings that does not sacrifice the asymmetries of the directed graph. Our approach, asymmetric node similarity embedding (ANSE), simultaneously learns the node embedding vectors and the parameters of an asymmetric similarity

function on the embeddings. This function allows the direction of edges to be recovered from the embeddings. Additionally we provide an adaptation to our method which embeds the graph onto a hypersphere. Like other random walk approaches, our method scales linearly with the size of the graph. We provide an illustration of a 2-dimensional embedding of a small lattice graph as well as demonstrate the effectiveness of our technique in several link prediction tasks on real world directed networks.

2 Problem Definition

A network, or graph, is used to represent objects and relationships. We define an undirected graph $G = (V, E)$ to be a set of vertices (nodes), $V = \{v_0, v_1, ..., v_{N-1}\}$, and a set of edges, $E = \{(v_i, v_j) \ : \ v_i, v_j \in V\}$, representing the objects and relationships respectively. A directed graph (digraph) imposes an ordering on the vertices of each edge such that (v_i, v_j) denotes an edge pointing from node v_i to node v_j. A random walk on a graph is a sequence of nodes $\left(v^{(0)}, v^{(1)}, ..., v^{(K)}\right)$ ordered such that $v^{(k+1)}$ is selected at random from the (outgoing) neighbors of $v^{(k)}$.

A node embedding is a function on a graph that maps each node in the graph to a d-dimensional vector $f_G : V \to \mathbb{R}^d$. The rows of the matrix $\boldsymbol{\Phi} \in \mathbb{R}^{|V| \times d}$ correspond to the vector embeddings of each node, i.e. $\boldsymbol{\Phi}_i = f_G(v_i)$. Define the similarity between two nodes $S(v_i, v_j)$ as proportional to the probability of visiting node v_j within k steps of a random walk beginning at node v_i. This function is asymmetric, $S(v_i, v_j) \neq S(v_j, v_i)$ for most complex networks. This is especially true for a directed graph in which $S(v_i, v_j) > 0 \implies S(v_j, v_i) > 0$.

Our goal is to preserve the asymmetric similarity between nodes in the embedded space. To this end, we aim to learn an embedding matrix $\boldsymbol{\Phi}$ as well as a similarity measure $K : \boldsymbol{\Phi} \times \boldsymbol{\Phi} \to \mathbb{R}$ such that $K(\boldsymbol{\Phi}_i, \boldsymbol{\Phi}_j) \propto S(v_i, v_j)$.

3 Skip-Gram Embedding

This section describes the skip-gram embedding model that forms the basis for random walk embedding methods including our approach and describes two algorithms that use skip-gram for embedding nodes in undirected graphs. The skip-gram method, first proposed for word embedding [6, 7] in natural language processing, extracts sentences from a document corpus and embeds each word to maximize the probability of predicting surrounding words. DeepWalk [10] and subsequent algorithms such Node2Vec [3] adapt the skip-gram model to node embedding for undirected graphs by substituting random walks for sentences.

Algorithm 1 outlines a model for skip-gram style node embedding methods. Lines 4 and 7 are the two key steps in the algorithm and consequently are where many node embedding methods differ from one another. Function $RandomWalk(G, v_i, k)$ on line 4 collects a sequence of k nodes on the graph from a random walk originating at vertex v_i. DeepWalk samples classical walks

while Node2Vec uses a random walk that can be weighted to remain close to the initial node or explore further away in the graph. In both methods, every pair of nodes in a walk within k-steps of one another are collected as positive samples.

The probability in line 7 of Alg. 1 is calculated using a softmax:

$$P(v_k|\boldsymbol{\phi}_j) = \frac{\exp\langle\boldsymbol{\phi}_j, \boldsymbol{\phi}_k\rangle}{\sum_l \exp\langle\boldsymbol{\phi}_j, \boldsymbol{\phi}_l\rangle}. \tag{1}$$

In practice this calculation is prohibitively expensive because it requires an inner product between the embeddings of the source node and each other node in the graph. Many algorithms employ alternative, less computationally expensive, methods of approximating the conditional probabilities. DeepWalk utilizes a hierarchical softmax [8] for computing probabilities while Node2Vec uses a negative sampling approach [7]. Negative sampling samples N node pairs from a noise distribution $P(v)$ as negative samples to approximates the log softmax as:

$$\log P(v_k|\boldsymbol{\phi}_j) \approx \log\left(\sigma\langle\boldsymbol{\phi}_j, \boldsymbol{\phi}_k\rangle\right) + \sum_{n=1}^{N} \mathbb{E}_{v_n\sim P(v)} \log(\sigma\langle-\boldsymbol{\phi}_j, \boldsymbol{\phi}_n\rangle) \tag{2}$$

where σ is the sigmoid function. The hierarchical softmax in DeepWalk reduces the complexity of each softmax calculation from $O(|V|)$ to $O(\log|V|)$ by using a binary tree to calculate the conditional probabilities. The negative sampling approach of Node2Vec reduces the complexity further to $O(N)$ where N is the number of negative samples.

Algorithm 1: Skip-Gram Model of Node Embedding

input : graph: G=(V,E)
window size: w
embedding size: d
walks per vertex: γ
walk length: k
output: node embeddings: $\boldsymbol{\phi}$

```
1  Initialize Φ ∈ ℝ^{|V|×d}
2  for i = 1 to γ do
3      for All nodes v_i ∈ V do
4          W_{v_i} = RandomWalk(G, v_i, k)
5          for v_j ∈ W_{v_i} do
6              for v_k ∈ W_{v_i}[j − w : j + w] do
7                  J(Φ) = − log P(v_k|φ_j)
8                  Φ = Φ − α ∗ ∂J/∂Φ
9              end
10         end
11     end
12 end
```

4 Method

Many symmetric properties of undirected graphs are asymmetric on digraphs. For example the graph geodesic, the length of the shortest path between nodes, is not symmetric for a directed network. The existence of a path from v_i to v_j does not even imply that a path exists for v_j to v_i.

To learn an embedding that can retain the asymmetric relationships between nodes, we replace the standard (symmetric) inner product used in the softmax (1) and negative sampling (2) equations with an asymmetric bilinear product defined by a matrix $\mathbf{A}$:

$$k_{\mathbf{A}}(v_i, v_j) = \langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle_{\mathbf{A}} = \boldsymbol{\phi}_i^T \mathbf{A} \boldsymbol{\phi}_j. \tag{3}$$

If $\mathbf{A}$ is not symmetric then in general $k_{\mathbf{A}}(v_i, v_j) \neq k_{\mathbf{A}}(v_j, v_i)$. Matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ can be learned in tandem with the embedding matrix $\boldsymbol{\Phi}$ through stochastic gradient descent. Geometrically, $\mathbf{A}$ can be viewed as defining a vector field that determines the direction of the most similar embedding vectors at any point.

Several challenges in sampling random walks on a digraph must be addressed that aren't present in undirected graphs. First, due to the directed nature of edges, positive node pair samples must remain ordered as they appear in the random walk. Second, walks in directed graphs may dead end. In a connected undirected graph a random walk can continue indefinitely by retracing the last edge the walk took to return from an otherwise terminal node. This is not the case for a connected digraph in which nodes may not have any outgoing edges. In such an event the walk is forced to terminate early.

A consequence of these two issues is that nodes without outgoing edges will never be sampled first in a pair. In these cases there is an increased importance in reaching the node from random walks beginning at other nodes so that positive samples containing the node are still collected. In scenarios in which a node has an arbitrarily small likelihood of being reached in a random walk, e.g. the only incoming edge to a node comes from another node with a high out-degree, the node is unlikely to appear as the second node in any positive pair. To address these issues we introduce a reverse random walk sampling method in addition to regular random walk sampling. Reverse walks are sampled from a dual graph, $G^* = (V, (v_j, v_i) : (v_i, v_j) \in E)$, where all edges are reversed. The sequence of nodes in the walk is then reversed again to provide a random walk on G whose transition probabilities are proportional to the in-degrees of nodes rather than their out-degrees. This guarantees that there are sample pairs containing each node with at least one incoming edge as the second node.

We use negative sampling to approximate the softmax function. Nodes are randomly sampled from the graph and used as negative samples as in (2). Combining the positive and negative sampling methods with the asymmetric similarity function produces our method given in Algorithm 2. We split the negative sampling into randomly sampling either the first or second nodes in the negative pairs, line 7 and line 9 respectively.

Algorithm 2: Asymmetric Node Similarity Embedding

input : graph: G(V,E)
embedding dimesnion: d
walks per vertex: γ
walk length: k
output: embeddings matrix: $\boldsymbol{\Phi}$
similarity matrix: $\mathbf{A}$

Initialize $\boldsymbol{\Phi} \in \mathbb{R}^{|V| \times d}$, $\mathbf{A} \in \mathbb{R}^{d \times d}$
for *All nodes* $v_i \in V$ **do**
  for $w = 1$ *to* γ **do**
    $W_{v_i} = RandomWalk(G, v_i, k)$
    $W_{v_i}^{rev} = ReverseRandomWalk(G^*, v_i, k)$
    $J_1(\boldsymbol{\Phi}) = -\sum_{v_j \in W_{v_i}} \log(\sigma \langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle_{\mathbf{A}})$
    $J_2(\boldsymbol{\Phi}) = -\sum_{n=1}^{N} \mathbb{E}_{v_n \sim P(v)} \log(\sigma \langle -\boldsymbol{\phi}_i, \boldsymbol{\phi}_n \rangle_{\mathbf{A}})$
    $J_3(\boldsymbol{\Phi}) = -\sum_{v_j \in W_{v_i}^{rev}} \log(\sigma \langle \boldsymbol{\phi}_j, \boldsymbol{\phi}_i \rangle_{\mathbf{A}})$
    $J_4(\boldsymbol{\Phi}) = -\sum_{n=1}^{N} \mathbb{E}_{v_n \sim P(v)} \log(\sigma \langle -\boldsymbol{\phi}_n, \boldsymbol{\phi}_i \rangle_{\mathbf{A}})$
    $J(\boldsymbol{\Phi}) = \sum_{n=1}^{4} J_n(\boldsymbol{\Phi})$
    $\boldsymbol{\Phi} = \boldsymbol{\Phi} - \alpha * \frac{\partial J}{\partial \boldsymbol{\Phi}}$
  end
end

4.1 Hypersphere Embedding

The embedding architecture can be adapted to embed nodes of a (di)graph onto the unit hypersphere. To do so we constrain the node embedding vectors to have unit length: $||\boldsymbol{\phi}_i||_2^2 = 1$. This is accomplished by renormalizing the length of the vector following each backpropagation update.

Matrix $\mathbf{A}$ should also be constrained such that $\forall v_i, v_j \in V : -1 \leq k_{\mathbf{A}}(v_i, v_j) \leq 1$. This constraint allows the similarity function to match the range of a standard inner product between two points on the unit hypersphere. If $\mathbf{A}$ is a unitary matrix, the product $\boldsymbol{\phi}_i^T \mathbf{A}$ will also have unit length and thus $-1 \leq \boldsymbol{\phi}_i^T \mathbf{A} \boldsymbol{\phi}_j \leq 1$ guaranteeing the constraint will hold.

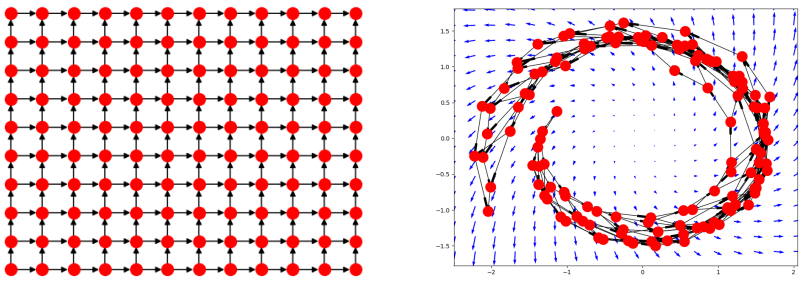
We can project $\mathbf{A}$ onto the set of unitary matrices whenever it diverges during learning similar to renormalizing the embedding vectors. Unfortunately, this projection is computationally costly, requiring a singular value decomposition of the matrix. Alternatively, we compose $\mathbf{A}$ as the product of a set of elementary reflector matrices of the form $\mathbf{A}^{(m)} = \mathbf{I} - 2 * \frac{\mathbf{v}_m \mathbf{v}_m^T}{\mathbf{v}_m^T \mathbf{v}_m}$ where $\mathbf{v}_m$ is any vector and $\mathbf{I}$ is the identity matrix. Any unitary matrix can be decomposed into a product of elementary reflectors. We use this decomposition to efficiently construct a unitary matrix $\mathbf{A} = \prod_{m=1}^{M} \mathbf{A}^{(m)}$ where M can chosen from 1 to the embedding dimension d. Smaller values of M restrict the the space of possible unitary matrices but also reduces both the computational cost to calculate $\mathbf{A}$ and the number of parameters for the model to learn. The vectors $\mathbf{v}_k$ are learned by backpropagating the loss through $\mathbf{A}$.

5 Experiments

In this section we conduct multiple experiments to demonstrate both quantitatively and qualitatively the effectiveness of our approach.

5.1 Lattice Example

We use a 2D lattice graph to provide a visual representation of our embedding scheme. The lattice is composed of 10 rows and 12 columns of vertices. All lateral edges in the graph are oriented to point right and all vertical edges to point up. Eight walks of length three are sampled from every node in the graph. The lattice is shown in Figure 1a and the learned embedding of the vertices in shown in 1b. Additionally the effect of the matrix $\mathbf{A}$ in the similarity function is drawn as a vector field in the background such that a source node is most similar to target nodes that lie in the direction of the local arrows. The embedded nodes form a spiral pattern with the bottom-left-most node of the original lattice innermost in the spiral and the top-right-most one outermost. Diagonal sets of nodes in the original lattice representation are structurally similar in the graph and are roughly clustered together along the spiral. Edges are also oriented along the direction of the field induced by $\mathbf{A}$.



(a) Original Lattice Graph (b) Embedded Lattice Graph

Fig. 1. The original lattice (a) and vector embeddings (b). The direction and magnitude of the vector field illustrates the bias of the similarity measure across the space.

5.2 Link Prediction

Node embeddings can be used to predict missing or future edges in a graph by measuring pairwise similarities. Node pairs with a high similarity score are more likely to form an edge. We evaluate the area under the receiver-operator curve (AUC) for several real world networks to evaluate our method and compare to several other skip-gram algorithms.

Arxiv [5] is a co-authorship network consisting of 5242 nodes representing authors and 28980 edges linking authors who have co-authored a paper together.

Arxiv is the only undirected network in this set of experiments. **Cora** [12] is a citation network where the 23166 nodes in the graph are papers and the 91500 edges points from one paper to another if the first paper cites the second. **Epinions** [11] is a social network dataset with 75879 users (nodes) and 508837 edges indicating trust placed by one user in another.

The reciprocity of a graph is the ratio of the number of bidirectional edges to the total number of edges. The three datasets we evaluate on vary wildly in reciprocity from Arxiv whose reciprocity as an undirected graph is 1.00 to Cora with nearly 0 bidirectional edges and a reciprocity of 0.05. Epinions sits in the middle at 0.40. Together, the three graphs provide a range of node-pair relations from bidirectional, to one-directional, to unrelated.

We use the same hyper-parameters for asymmetric node similarity embeddings (ANSE) and the hypersphere variant (ANSE-H) across all three experiments. Nodes are embedded into a 32-dimensional space and 16 total walks (8 forward, 8 backward) are collected at each node. Each walk continues for 3 steps. For ANSE, we also clip the length of the embedding vectors to be less than 1. In ANSE-H, we use a single elementary reflector matrix for $\mathbf{A}$. We evaluate our method against several modern skip-gram style embedding methods. Deepwalk [10] and Node2Vec [3] are methods developed for undirected graphs. Line [13] and asymmetric proximity preserving embeddings (APP) [16] are methods for embedding nodes of either undirected or directed graphs. We compare ANSE against the scores for the other methods reported in [16]. We also compare against HOPE [9] a recent spectral method for directed graphs that learns both a source and target embedding for each node. We also using a 32 dimensional embedding for HOPE.

The embedding methods are trained using 70% of the edges of the graph while the remaining 30% appear as positive examples in the test set along with an equal number of random node pairs without edges to form the negative examples. The pairwise node score is used to predict the existence of an edge or not between each pair of nodes in the test set. The AUC for each method is given in Table 1. On two of the three graph domains ANSE and ANSE-H demonstrates a significant improvement over all the other methods tested. On Cora, APP joins ANSE and ANSE-H in outperforming the other methods tested. Despite Arxiv being an undirected graph our method learns the asymmetric random walk similarities between pairs of nodes resulting in the high AUC.

6 Related Work

Several node embedding methods for digraphs embed the nodes twice, once into a source space and a second time into a target space [4, 9, 16]. APP [16] is a skip-gram model that learns dual embeddings for each node in the graph. Node pair samples are collected from the endpoints of (directed) random walks. These pairs are ordered so that the similarity score between nodes is calculated using the source embedding of the first node and the target embedding of the second.

Table 1. Link Prediction Area Under Curve (AUC)

Network	Arxiv	Cora	Epinions
DeepWalk	0.887	0.936	0.823
Node2Vec	0.810	0.734	0.865
Line	0.750	0.694	0.867
APP	0.887	**0.944**	0.926
HOPE	0.596	0.874	0.629
ANSE	0.902	**0.941**	**0.948**
ANSE-H	**0.920**	**0.942**	0.924

Our embedding technique can also be viewed as learning a dual embedding where the source embedding for node v_i is $\boldsymbol{\phi}_i$ and the target embedding is $\mathbf{A}\boldsymbol{\phi}_i$. Compared to learning two embeddings, our approach reduces the number of embedding parameters from $2|V|d$ to $|V|d + d^2$, as typically $d \ll |V|$. Additionally, tying the source and target embeddings together in our approach overcomes a potential issue in [16] in which the source or target embeddings of a node in a digraph may have no positive samples if the node has no outgoing or incoming edges respectively.

A related asymmetric bilinear product is used in [14] for text retrieval. The asymmetric Hermitian inner product is used to score co-attention between complex valued word vectors. The bilinear product $s_{ij} = Re(a_i^T \mathbf{M} b_j)$ is also studied. Unlike our work, however, the matrix $\mathbf{M}$ was tuned as a hyperparameter rather than allowed to change during learning.

7 Conclusion

In this paper, we proposed ANSE, a scalable method to embed a digraph into a vector space, and ANSE-H, a variant of ANSE that embeds the graph onto a hypersphere. ANSE simultaneously learns a vector representations of nodes and an asymmetric similarity function for embedding directed networks. Learning both the embedding and the similarity function offers the ability to recover the direction of edges from the embedded nodes. Additionally we proposed a random walk sampling method to improve learning for nodes without either incoming or outgoing edges. On multiple real world datasets, ANSE and ANSE-H outperforms other skip-gram embedding schemes for link prediction.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation **15**(6), 1373–1396 (2003). https://doi.org/10.1162/089976603321780317
2. Coifman, R.R., Lafon, S.: Diffusion maps. Applied and computational harmonic analysis **21**(1), 5–30 (2006)

3. Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 855–864. KDD '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939754, http://doi.acm.org/10.1145/2939672.2939754
4. Khosla, M., Leonhardt, J., Nejdl, W., Anand, A.: Node representation learning for directed graphs. ArXiv **abs/1810.09176** (2018)
5. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM Transactions on Knowledge Discovery from Data (TKDD) **1**(1), 2 (2007)
6. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (2013)
7. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
8. Mnih, A., Hinton, G.E.: A scalable hierarchical distributed language model. In: Advances in neural information processing systems. pp. 1081–1088 (2009)
9. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1105–1114. KDD '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939751, http://doi.acm.org/10.1145/2939672.2939751
10. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 701–710. KDD '14, ACM, New York, NY, USA (2014). https://doi.org/10.1145/2623330.2623732, http://doi.acm.org/10.1145/2623330.2623732
11. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In: International semantic Web conference. pp. 351–368. Springer (2003)
12. Šubelj, L., Bajec, M.: Model of complex networks based on citation dynamics. In: Proceedings of the 22nd international conference on World Wide Web. pp. 527–530. ACM (2013)
13. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. pp. 1067–1077. International World Wide Web Conferences Steering Committee (2015)
14. Tay, Y., Luu, A.T., Hui, S.C.: Hermitian co-attention networks for text matching in asymmetrical domains. In: IJCAI. pp. 4425–4431 (2018)
15. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. science **290**(5500), 2319–2323 (2000)
16. Zhou, C., Liu, Y., Liu, X., Liu, Z., Gao, J.: Scalable graph embedding for asymmetric proximity. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)