# Exploiting Gaussian Embeddings for Directed Link Prediction*

Inzamam Rahaman
The University of the West Indies
Department of Computer Science
St. Augustine, Trinidad and Tobago
inzamam@lab.tt

Patrick Hosein
The University of the West Indies
Department of Computer Science
St. Augustine, Trinidad and Tobago
patrick.hosein@sta.uwi.edu

## ABSTRACT

Many systems can be represented as graphs where nodes represent components or actors in the system and edges represent interactions between said components and actors. Link Prediction refers to the task of using the current graphical representation of a system to suggest which interactions might not be captured by the system or which interactions might occur in the future. In this position paper, we present preliminary results on the usage of Gaussian embeddings of nodes for the task of link prediction in directed graphs.

## CCS CONCEPTS

• **Computing methodologies → Machine learning**; **Classification and regression trees**;

## KEYWORDS

Distributed Embedding, Link Prediction

**ACM Reference format:**
Inzamam Rahaman and Patrick Hosein. 2017. Exploiting Gaussian Embeddings for Directed Link Prediction. In *Proceedings of The 13th International Workshop on Mining and Learning on Graphs, Halifax, Nova Scotia, Canada, Aug 2017 (MLG' 17),* 3 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Many social, biological, and physical phenomenon can be represented using graphs where a set of nodes - usually denoted $V$ - represent discrete components or actors and edges - usually denoted as $E$, where $E \subseteq V \times V$ - represent interactions between said components or actors. For example, social networks, such as Facebook, can be represented as graphs where nodes represent users and an edge between two users represents that said two users are friends. These graphs can be directed or undirected, weighted or unweighted, heterogeneous or homogeneous.

Given the representation of a system as a graph, we often wish to predict the evolution of the graph, determine anomalies in the graph, or impute missing edges between nodes. This task is known as link prediction. Link prediction can be useful in many different contexts. In e-commerce systems, we can suggest products to customers [5]. In a social network, we can use link prediction to suggest potential friends [2].

Much work has been done on link prediction in undirected graphs [1], [8], [7], [3], [10]. In an undirected graph, if $(u, v) \in E$, then $(v, u) \in E$.

However, in many systems, such as Twitter or scientific citation networks, interactions are unidirectional. For example, in Twitter, if user $u$ follows user $v$, $v$ does not necessarily follow $u$. Consequently, such systems must be represented by directed graphs. Many methods for link prediction in undirected graphs tend to use features that are symmetric. i.e., if $f$ represents the method for extracting features related to a pair of nodes, $f(u, v) = f(v, u)$. This symmetry might assist in determining whether an edge exists between two nodes or not, but might not assist much in determining the direction of the edge. For example, suppose that in some graph, $G = (V, E)$ there is an edge going from $u$ to $v$. Suppose then, for notational purposes, we say that $(u, v)$ exists in $G$ ($(u, v)$ is a positive example), but that $(v, u)$ does not ($(v, u)$ is a negative example). A classifier using a symmetric feature might incorrectly indicate that both $(u, v)$ and $(v, u)$ belong in the positive class when it ought to output that $(u, v)$ belongs in the positive class and $(v, u)$ belongs in the negative class. Consequently, the task of directed link prediction would benefit from features that are asymmetric with respect to node pairs.

## 2 METHOD

Vilnis and McCallum [9] showed that embedding words from a corpus in a space of multivariate Gaussian distributions might better capture uncertainties and assist in tasks where asymmetric relationships between words might be important. Vilnis and McCallum used an approach similar to that used to derive distributed representations in word2vec [6]. In training the Gaussian embeddings, Vilnis and McCallum optimized the negation of the KL divergence between pairs of distributions as described in the equations:

$$E(u, v) = -D_{KL}(\mathcal{N}_u || \mathcal{N}_v) \tag{1}$$

$$D_{KL}(\mathcal{N}_u || \mathcal{N}_v) = \frac{1}{2}(tr(\Sigma_u^{-1}\Sigma_v) + (\mu_u + \mu_v)^T \Sigma_u^{-1}(\mu_u + \mu_v) - d - log\frac{\Sigma_u}{\Sigma_v}) \tag{2}$$

where $u$ and $v$ are words, $tr$ is the trace of the matrix, $\mathcal{N}_i$ is the Gaussian embedding for word $i$, $\mu_i$ is the mean vector of the Gaussian embedding for word $i$, and $\Sigma_i$ is the covariance matrix of the Gaussian embedding for word $i$. Much like in word2vec [6] Vilnis and McCallum used parameters to define the window size about a word in the corpus for sampling and to define the dimensionality of the learned Gaussian distributions.

*Position paper

In addition, Grover and Leskovec [3] developed a method of learning representations of nodes using alias sample-guided random walks. In Grover and Leskovec's [3] method, alias sampling for the random walks was guided by a return parameter, $p$, that biases the random walk to return to previously visited nodes, and an inwards-outwards parameter, $q$, that mediates how far the walk strays from its starting node. Using this approach Grover and Leskovec's method can interpolate between depth-first and breath-first strategies for navigating the graph [3]. Using a defined walk length and number of walks per starting node, these random walks would generate a corpus that would then be used in a word2vec [6] based approach of using stochastic gradient descent to learn embeddings for the word within a real vector space of a specified number of dimensions.

In our work, we used the random walk sampling approach described by Grover and Leskovec [3] to generate a corpus where nodes, linked to unique identifiers, behaved as words, and then used the energy based learning method described by Vilnis and McCallum [9] to train multivariate Gaussian embeddings. Using these trained embeddings, we then constructed a feature vector as follows:

$$f(u, v) = [D_{KL}(\mathcal{N}_u||\mathcal{N}_v), D_{KL}(\mathcal{N}_v||\mathcal{N}_u)] \tag{3}$$

where $u$ and $v$ are the nodes under consideration, and $\mathcal{N}_i$ is the Gaussian embedding for node $i$. Since the KL divergence is asymmetric, both elements of this feature vector would be different for all pairs of nodes where $u \neq v$. These feature vectors are then used as input to an appropriate classifier for link prediction. It should be noted that once embeddings have been trained, the process of computing the aforementioned feature vector can be easily parallelized.

## 3 EXPERIMENTS

In addition to our KL-divergence vector feature representation for nodes, we also evaluate the performance of several other feature representations: the Adamic-Adar Coefficient [1], Jaccard's coefficient [8], Resource Allocation Index [10], and the Hadamard product of the learned node2vec distributed representations [3]. In addition, we also evaluated the performance of the Hadamard product node2vec representation augmented with our KL-divergence featured representation; this is denoted as "node2vec Hadarmard + KL divergence vector" in Tables 2, 3, and 4. All of these constructed feature representations were used as inputs into logistic regression classifiers.

Note that both Grover and Leskovec's word2vec [3] and our method use many of the same parameters. Consequently, we fixed $p$ to 0.5, $q$ to 0.5, window size to 10, number of walks to 100, walk length to 100 and the number of dimensions to 64 for both node2vec and our method. In addition, our method, like Vilnis and McCallum's word2gauss [9] can also consider spherical and diagonal covariances; we used diagonal covariances for all of our experiments.

### 3.1 Datasets

To evaluate our methods, we used several datasets provided by the Stanford Network Analysis Project [4]. These datasets and their properties are described in Table 1.

**Table 1: Description of Datasets**

| Dataset | Number of Nodes | Number of Edges |
|---|---|---|
| Wikivote | 7115 | 103689 |
| Wiki-Rfa | 11217 | 188837 |
| cit-HepTh | 27770 | 352807 |

**Table 2: Performance of Methods on Wikivote Dataset**

| Method | microF1 | AUC |
|---|---|---|
| Resource Allocation Index | 0.62 | 0.74 |
| Adamic-Adar Coefficient | 0.66 | 0.75 |
| Jaccard Coeffiecient | 0.66 | 0.75 |
| node2vec hadamard | 0.74 | 0.88 |
| KL divergence vector | 0.82 | 0.90 |
| node2vec hadamard + KL divergence vector | 0.84 | 0.92 |

**Table 3: Performance of Methods on Wiki-Rfa Dataset**

| Method | microF1 | AUC |
|---|---|---|
| Resource Allocation Index | 0.65 | 0.76 |
| Adamic-Adar Coefficient | 0.68 | 0.76 |
| Jaccard Coeffiecient | 0.69 | 0.76 |
| node2vec hadamard | 0.90 | 0.96 |
| KL divergence vector | 0.78 | 0.87 |
| node2vec hadamard + KL divergence vector | 0.91 | 0.97 |

**Table 4: Performance of Methods on cit-HepTh Dataset**

| Method | microF1 | AUC |
|---|---|---|
| Resource Allocation Index | 0.76 | 0.81 |
| Adamic-Adar Coefficient | 0.77 | 0.81 |
| Jaccard Coeffiecient | 0.74 | 0.81 |
| node2vec hadamard | 0.73 | 0.83 |
| KL divergence vector | 0.80 | 0.88 |
| node2vec hadamard + KL divergence vector | 0.82 | 0.91 |

For each dataset, we randomly selected 50% of the edges (positive examples) to use for training and 50% to use for testing. After extracting the positive examples for both the training set and testing set, we generated an equal number of negative examples, i.e., pairs of nodes for which no directed edge exists in the original dataset, such that there was no overlap of examples between the training set and testing set for each dataset.

### 3.2 Results

After training each classifier using the training set, we evaluated the performance on the appropriate testing set using microF1 and AUC scores. These results can be seen in Tables 2, 3, and 4.

As can be seen in Tables 2, 3, and 4, the embedding based methods (both ours and Grover and Leskovec's) perform better than the heuristic score methods. Furthermore, the heuristic score methods tended to have similar performance across all datasets. Moreover, the "node2vec Hadamard + KL divergence vector" was the best performing method across all datasets. Our KL divergence vector method also outperformed Grover and Leskovec's method on two of the three datasets. Determining what features of the Wiki-Rfa dataset that facilitated this would be an interesting avenue for future work. Interestingly, despite the Hadamard product of the learned node2vec embeddings being commutative, and as a consequence, symmetric, Grover and Leskovec's method performed fairly well on these directed graph datasets, and as aforementioned, was the second-best performing method for the Wiki-Rfa dataset.

## 4 DISCUSSION

In this paper, we motivated the usage of asymmetric features for link prediction in directed graphs. In addition, proceeding in a manner similar to Grover and Leskovec [3], we adapted a method for embedding words in a corpus to function on graphs. We then showed that our proposed KL divergence vector feature extraction method can be useful for training a classifier for link prediction in directed graphs.

Among the avenues for future work would be to explore reasons for node2vec's performance relative to our method's performance on the Wiki-Rfa dataset. Furthermore, we plan to investigate further applications of Gaussian embeddings in the space of graph mining and learning. In addition, as noted by Vilnis and McCallum, the space of a multimodal distribution might be an interesting target for generating embeddings.

## ACKNOWLEDGMENTS

The authors would like to thank the Trinidad and Tobago Network Information Centre for providing funding for this research. Furthermore, we would like to thank the anonymous peer reviewers for their comments.

## REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social networks* 25, 3 (2003), 211–230.

[2] Luca Maria Aiello, Alain Barrat, Rossano Schifanella, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. 2012. Friendship prediction and homophily in social media. *ACM Transactions on the Web (TWEB)* 6, 2 (2012), 9.

[3] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[4] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

[5] Xin Li and Hsinchun Chen. 2013. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems* 54, 2 (2013), 880–890.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[7] Mark EJ Newman. 2001. Clustering and preferential attachment in growing networks. *Physical review E* 64, 2 (2001), 025102.

[8] Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

[9] Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623* (2014).

[10] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. 2009. Predicting missing links via local information. *The European Physical Journal B-Condensed Matter and Complex Systems* 71, 4 (2009), 623–630.