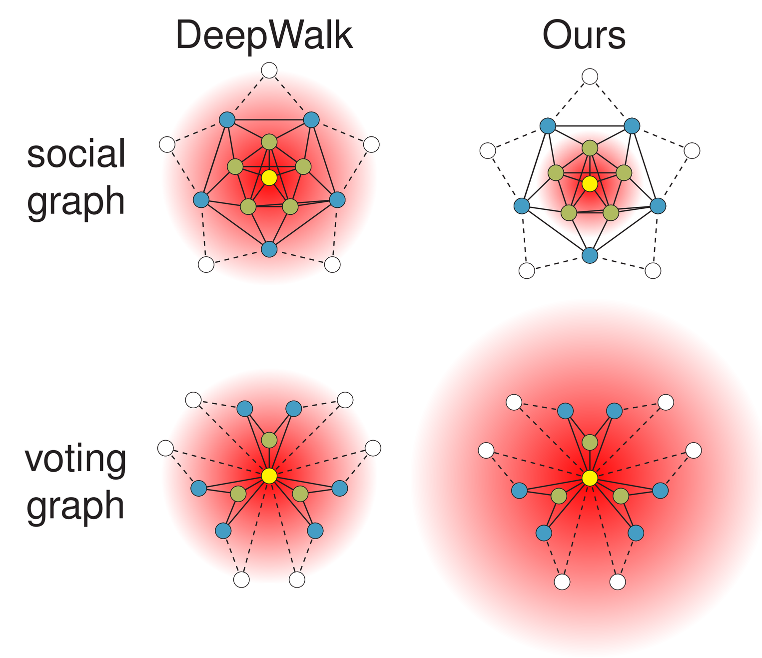
Watch Your Step: Learning Node Embeddings via Graph Attention

Sami Abu-El-Haija‡, Bryan Perozzi†, Rami Al-Rfou†, Alex Alemi†
‡ USC ISI, † Google AI (✉: sami@haija.org, bperozzi@acm.org)

Overview

- **Task.** *Embedding a Graph*: mapping nodes onto a d-dimensional continuous vector space.
- **Why?** Continuous Representations can then be used for task-specific ML models (e.g. Link Prediction or Node Classification).
- **Motivation.** Embedding methods based on Random Walks [2] produce powerful representations. However, they operate in two discrete steps (Random Walks then Representation Learning), and contain hyper-parameters (e.g. walk length) that must be tuned per graph.
- **Our Contribution.** We replace previously-fixed hyper-parameters with trainable parameters that we automatically tune by back-propagation while jointly learning node embeddings.
- **Method.** The hyper-parameters impose a distribution on every node's neighbourhood, which we term *context distribution* and denote Q. We learn Q that best-preserves the graph structure. We parametrize Q as an attention model on the power series of the graph transition matrix.
- **Results.** Our method significantly improves performance on Link Prediction by 20%-40% for all graphs. Further, the automatically-learned context distribution agrees with the optimal hyper-parameter choices, if we manually tune existing methods.



Problem Statement

- Given a graph $G = (V, E)$, an embedding algorithm produces matrix $\mathbf{Y} \in \mathbb{R}^{|V| \times d}$ with row Y_u being the d-dimensional (embedding) representation for node $u \in V$.
- Embeddings should preserve the structure of the graph: two node embeddings should be close if they are neighbors.
- Quality of embeddings can be measured on link-prediction tasks, as it is desirable to generalize to unseen information.

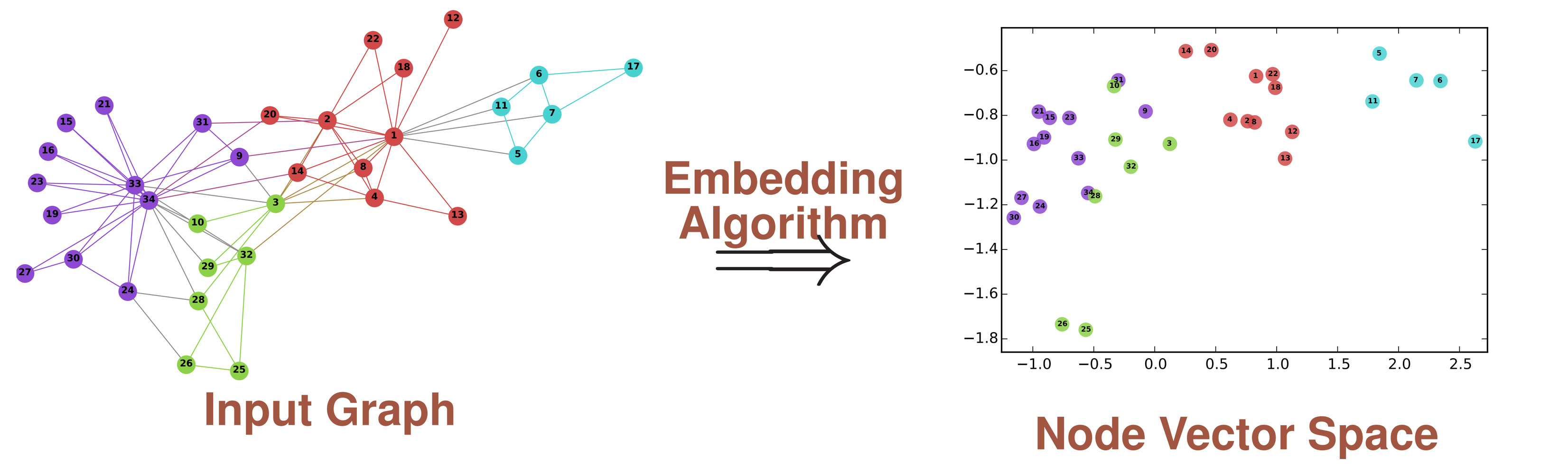
Classical Approach

Earlier approaches to Node Embeddings include Laplacian Eigenmaps [1]:

$$\min_{\mathbf{Y}} \sum_{(u,v) \in E} ||Y_u - Y_v||_2^2, \quad (1)$$

Solved as eigendecomposition of graph Laplacian matrix, which avoids trivial solutions and is equivalent to applying orthonormality constraints: $\mathbf{Y} \text{diag}(\vec{1}^\top \mathbf{A}) \mathbf{Y}^\top = I$.

2D Embedding of Karate Club Network [2]:

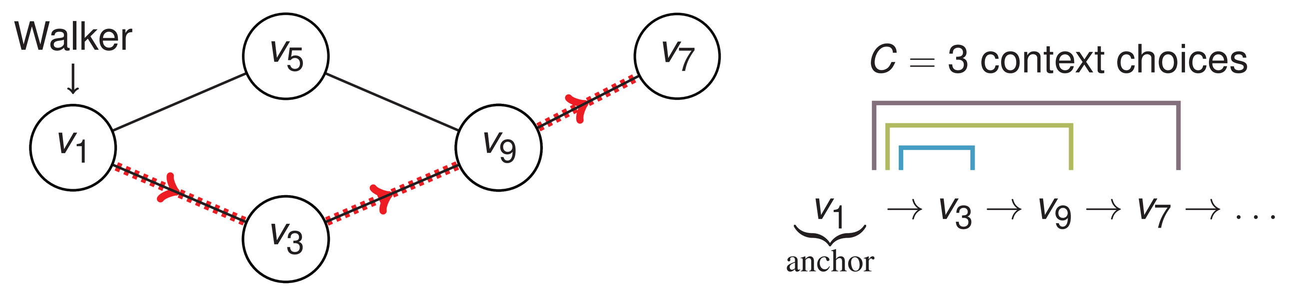


Review: Embedding via Random Walks

Introduced by Perozzi et al [2], this family of algorithms (including AsymProj[3], node2vec[4]):

- Operate in two disjoint steps of (i) Random Walk simulation; (ii) Representation Learning.
- Each of the steps has hyper-parameters
- Step (ii) is done by training a Skipgram model (from word2vec [5]) over the walk sequences.

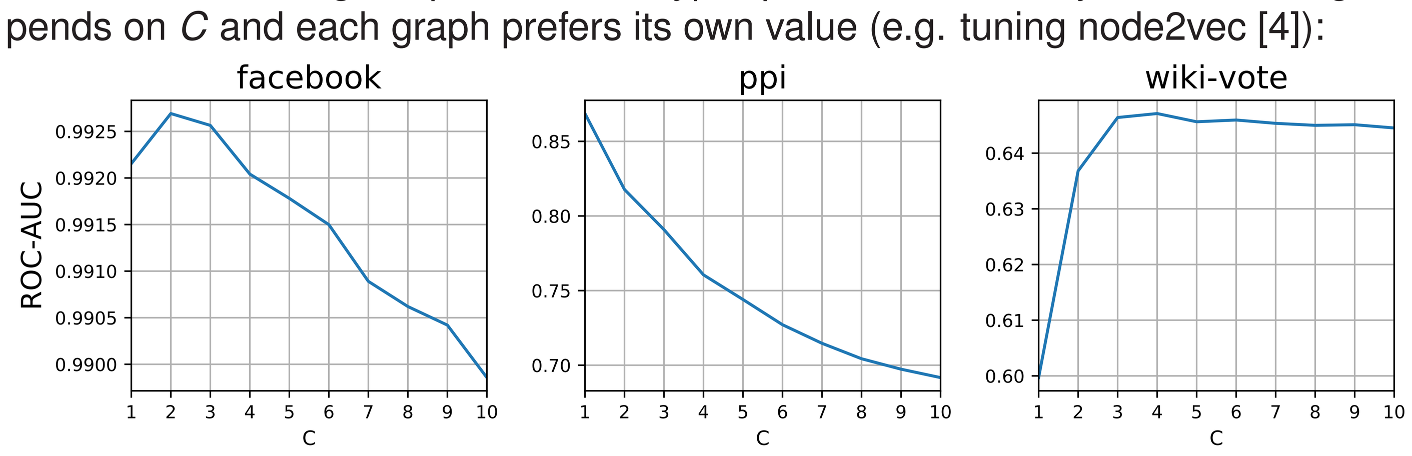
Skipgram Context in Graphs (as used by DeepWalk, n2v, etc):



- Depicted for context size $C = 3$. At each (anchor) node along sequence, a coin is flipped uniformly: $c \sim \mathcal{U}\{1, C\}$, which determines the context nodes that get selected.
- Embedding of Anchor is brought closer to the context.
- The sampling process (rather than fixing $c = C$) is crucial per [6].

Weaknesses of Existing Methods:

- One must do a large exploration on hyper-parameters Quality of embeddings heavily depends on C and each graph prefers its own value (e.g. tuning node2vec [4]):



- Even though C can be manually-tuned, most of these methods use word2vec implementation and therefore inherit the context sampling: $c \sim \mathcal{U}\{1, C\}$.

Deriving our Method: Embedding via Matrix Factorization

- The random walk simulation, context sampling ($c \sim \mathcal{U}\{1, C\}$) and representation learning, can all be replaced by factorizing node-to-node co-occurrence matrix (similar to [7]).
- Let $\mathbf{D} \in \mathbb{R}^{N \times N}$ be a node-to-node where D_{uv} counts the event of u appearing c-steps after v (with $c \sim \mathcal{U}\{1, C\}$) across all random walks.

Objective Function: We factorize $\mathbf{D}$ using negative-log *graph likelihood* objective of [3], written in our notation:

$$\min_{\mathbf{L}, \mathbf{R}} \left|\left| -\mathbf{D} \circ \log\left(\sigma(\mathbf{L} \times \mathbf{R}^\top)\right) - 1[\mathbf{A} = 0] \circ \log\left(1 - \sigma(\mathbf{L} \times \mathbf{R}^\top)\right)\right|\right|_1, \quad (2)$$

Where nodes are embedded into two (asymmetric) embedding spaces $\mathbf{L}, \mathbf{R} \in \mathbb{R}^{N \times \frac{d}{2}}$ (i.e. $\mathbf{Y} = [\mathbf{L}|\mathbf{R}]$) and the pairwise edge scoring model is their outer-product. Indicator function $1[.]$ is applied element-wise. L_1 norm of the matrix is sum of its entries, which are all positive since element-wise standard logistic $\sigma : \mathbb{R} \rightarrow (0, 1)$

$\mathbb{E}[\mathbf{D}]$ and Context Distribution Q

- Context Distribution Q assigns higher mass to nearby nodes, but the specific form of Q depends on hyper-parameters (e.g. C and choice of $\mathcal{U}$). The value of Q affects values in the node-to-node matrix $\mathbf{D}$.
- As derived in our Appendix, with DeepWalk [2], $\mathbb{E}[\mathbf{D}]$ can be written as:

$$\mathbb{E}\left[\mathbf{D}^{\text{DEEPWALK}}; C\right] = \tilde{\mathbf{P}}^{(0)} \sum_{k=1}^{C} \left[1 - \frac{k-1}{C}\right] (\mathcal{T})^k, \quad (3)$$

where $\tilde{\mathbf{P}}^{(0)}$ is a diagonal matrix containing the number of walks to be started from each node. We set the diagonal entries to 80.

- If GloVe [7] context sampling was used, we derive:

$$\mathbb{E}\left[\mathbf{D}^{\text{GloVe}}; C\right] = \tilde{\mathbf{P}}^{(0)} \sum_{k=1}^{C} \left[\frac{1}{k}\right] (\mathcal{T})^k. \quad (4)$$

- We want to learn the coefficients to $(\mathcal{T})^k$. We propose the parametrized expectation:

$$\mathbb{E}[\mathbf{D}; \mathbf{q}] = \tilde{\mathbf{P}}^{(0)} \sum_{k=1}^{C} Q_k (\mathcal{T})^k, \quad (5)$$

with:

$$Q_1, Q_2, \cdots = \text{softmax}(q_1, q_2, \ldots) \text{ and } q_k \in \mathbb{R}. \quad (6)$$

- Our final objective extends Graph Likelihood 2 with attention parameters

$$\min_{\mathbf{L}, \mathbf{R}, \mathbf{q}} \left|\left| -\mathbb{E}[\mathbf{D}; \mathbf{q}] \circ \log\left(\sigma(\mathbf{L} \times \mathbf{R}^\top)\right) - 1[\mathbf{A} = 0] \circ \log\left(1 - \sigma(\mathbf{L} \times \mathbf{R}^\top)\right)\right|\right|_1, \quad (7)$$

is minimized w.r.t. node embeddings $\mathbf{L}$, $\mathbf{R}$ and attention logit vector $\mathbf{q}$ (parametrizes Q)

- Attention parameters $\mathbf{q}$ can be thrown-away after training, as they are not part of the model and are not used for inference.

Experiment and Results

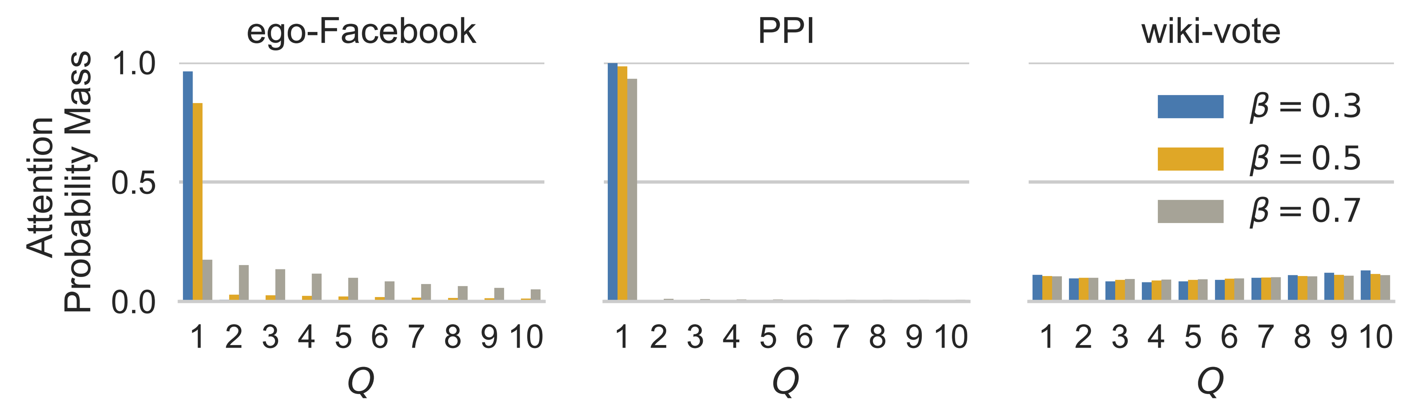
Link Prediction

- **Datasets:** We use the data splits of [3]. *wiki-vote* is voting network of Wikipedia. *ego-Facebook* is a social network. *ca-AstroPh* and *ca-HepTh* are citation networks. *PPI* is protein-protein interactions network.
- **Baselines**: Laplacian *EigenMaps* [1], Singular Value Decomposition (*SVD*) on adjacency matrix, *DNGR* is a deep auto-encoder network, node2vec (*n2v*) [4] with two C values (full C sweep is on left), and AsymProj is [3].

Results

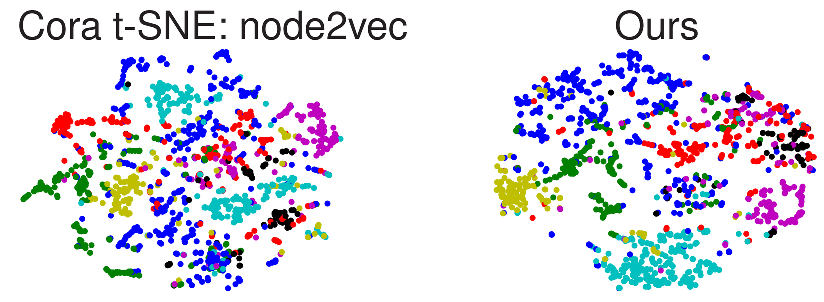
		Methods Use: A			D			$\mathbb{E}[\mathbf{D}]$	Error
Dataset	dim	Eigen Maps	SVD	DNGR	n2v $C=2$	n2v $C=5$	Asym Proj	Graph Attention *(ours)*	Reduction
wiki-vote	64	61.3	86.0	59.8	64.4	63.6	91.7	**93.8** $\pm$ **0.13**	25.2%
	128	62.2	80.8	55.4	63.7	64.6	91.7	**93.8** $\pm$ **0.05**	25.2%
ego-Facebook	64	96.4	96.7	98.1	99.1	99.0	97.4	**99.4** $\pm$ **0.10**	33.3%
	128	95.4	94.5	98.4	99.3	99.2	97.3	**99.5** $\pm$ **0.03**	28.6%
ca-AstroPh	64	82.4	91.1	93.9	97.4	96.9	95.7	**97.9** $\pm$ **0.21**	19.2%
	128	82.9	92.4	96.8	97.7	97.5	95.7	**98.1** $\pm$ **0.49**	24.0%
ca-HepTh	64	80.2	79.3	86.8	90.6	91.8	90.3	**93.6** $\pm$ **0.06**	22.0%
	128	81.2	78.0	89.7	90.1	92.0	90.3	**93.9** $\pm$ **0.05**	23.8%
PPI	64	70.7	75.4	76.7	79.7	70.6	82.4	**89.8** $\pm$ **1.05**	43.5%
	128	73.7	71.2	76.9	81.8	74.4	83.9	**91.0** $\pm$ **0.28**	44.2%

- Visualizing automatically-learned Q distribution:



Unsupervised Node Classification:

Dataset	n2v $C=5$	Graph Attention (ours)
Cora	63.1	**67.9**
Citeseer	45.6	**51.5**



References

[1] Belkin & Niyogi, *Laplacian Eigenmaps*, Neural Computation 2003.
[2] Perozzi et al, *DeepWalk*, KDD 2014
[3] Abu-El-Haija et al, *Edge Representation*, CIKM 2017
[4] Grover & Leskovec, *node2vec*, KDD 2016
[5] Mikolov et al, *word2vec*, NIPS 2013
[6] Levy et al, *Improving Distributional Sim.*, TACL 2015
[7] Pennington et al, *GloVe*, EMNLP 2014

Source code available at: `http://sami.haija.org/graph/context`