STABLE: Identifying and Mitigating Instability in Embeddings of the Degenerate Core

David Liu* Tina Eliassi-Rad†

Abstract

Are the embeddings of a graph's degenerate core stable? What happens to the embeddings of nodes in the degenerate core as we systematically remove periphery nodes (by repeatedly peeling off k-cores)? We discover three patterns w.r.t. instability in degenerate-core embeddings across a variety of popular graph embedding algorithms and datasets. We correlate instability with an increase in edge density, and then theoretically show that in the case of Erdös-Rényi graphs embedded with Laplacian Eigenmaps, the best and worst possible embeddings become less distinguishable as density increases. Furthermore, we present the `STABLE` algorithm, which takes an existing graph embedding algorithm and makes it stable. We show the effectiveness of `STABLE` in terms of making the degenerate-core embedding stable and still producing state-of-the-art link prediction performance.

1 Introduction

Previous work has presented varied evidence for the effectiveness of graph (a.k.a. node) embedding algorithms. For instance, while some suggest that graph embeddings improve performance on link prediction and node classification, others have shown that basic heuristics can outperform graph embeddings in community detection [25]. Other work has shown that the low dimensionality of embeddings prevents them from capturing the triangle structure of real-world networks [23]. In this work, we examine the stability of graph embeddings as a means for better understanding the information they capture and their utility in different contexts.

We measure the stability of the graph's degenerate core (i.e., its k-core with maximum k) as outer k-shells (i.e., the "periphery") are iteratively shaved off. The k-core of an undirected graph G is the maximal subgraph of G in which every node is adjacent to at least k nodes. A common approach to understanding stability is to measure changes to algorithmic output due to input perturbations. K-core analysis gives us a principled mechanism for changing graphs. In analyzing the embedded k-cores, we ask whether the embeddings capture the degenerate core's structure and if/how its embedding changes as each shell is removed. For example, previous work showed that degenerate cores are generally not cliques but contain community structure [24]. We study whether such patterns are preserved in the embeddings of the degenerate cores as k-shells are removed.

It is important to evaluate the stability of embeddings of nodes in the degenerate-core (a.k.a. "degenerate-core embeddings") because dense subgraphs are the "heart" of the graph. Nodes in the degenerate core are often the most influential spreaders. In marketing applications, the removal of a dense core node can trigger a cascade of node removals [15, 13]. Yet, as important as the core nodes are, previous studies on online activism have shown that core nodes are also dependent on periphery nodes to amplify messages originating from the core nodes [3]. In this study, we assess the importance of the periphery nodes in the stability of the core node embeddings – specifically, the nodes in the degenerate core.

As we present in this work, the embeddings of nodes in the degenerate core are not stable (as in they do not persist as the periphery is removed). Thus, graph embeddings are relative and not absolute. These possible perturbations are a concern because real-world networks are noisy [18, 29] and dynamic [12]. As such, unstable embeddings should push us to place less faith in any individual set of graph embeddings. Instead, we must better specify the noise in the network to qualify the quality of graph embeddings.

Our main contributions are as follows:

1. Across multiple categories of graphs and embedding algorithms, we discover three patterns of instability in the embeddings of nodes in the degenerate core. In the process, we introduce a method called `SHARE` for measuring the stability of degenerate-core embeddings.
2. We show that the instability in degenerate-core embeddings is correlated with increases in graph edge density when the periphery is removed. Subsequently, we theoretically analyze the specific case of embedding Erdös-Rényi graphs with Laplacian

*Northeastern University (liu.davi@northeastern.edu).
†Northeastern University (t.eliassirad@northeastern.edu).

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Eigenmaps and show that the performance gap between the best and worst embeddings narrows as the graph density increases.

3. We present an algorithm `STABLE`[1] for generating core-stable graph embeddings. Our algorithm is flexible enough to augment any existing graph embedding algorithm that minimizes a differentiable loss function. We show that when instantiated with Laplacian Eigenmaps [4] and LINE [26], our algorithm yields embeddings that preserve downstream utility while increasing stability.

2 Defining and Motivating Embedding Instability

We analyze the stability of graph embedding algorithms by tracking the embedding of the degenerate core (the k-core with maximal k), as we progressively shave k-shells.[2] After each shell is removed, we re-embed the remaining subgraph; we name this method SHave-And-Re-Embed, or "`SHARE`". Figure 1 illustrates the application of `SHARE` on the Zachary Karate Club graph [30]. In the figure, the degenerate core ($k = 4$) is highlighted in blue in the top row, and the bottom row plots the Node2Vec embedding for the remaining subgraph, where the corresponding degenerate-core embeddings are also plotted in blue. For the Karate Club graph, the relative proximities of the core embeddings do not change until the degenerate core is embedded in isolation, which we call the *isolated embeddings*.

We define stability as the property of being resilient to perturbation, a definition of stability that is common in the complex networks literature [28]. This definition contrasts stability with robustness, which is an insensitivity to microscopic changes across different settings or environments. In the case of core embeddings, we define stability as the resilience of the proximities of degenerate core embeddings when the periphery is perturbed.

To quantify the stability of degenerate-core embeddings, we use `SHARE` to measure the evolution of the distribution of pairwise distances in the embedded space, which we call the *degenerate-core pairwise distribution (DCPD)*. Specifically, `SHARE` begins with the entire graph ($k = 0$), embeds the nodes, and calculates the distribution of pairwise distances among the embeddings for the degenerate-core nodes. Next, `SHARE` takes the $k = 1$ core, re-embeds the nodes in the subgraph, and re-calculates the pairwise distribution for the degenerate-core nodes using the updated embeddings. If the embeddings are stable, the pairwise distributions should not vary as the k-shells are removed, as a change in the distribution would suggest perturbations to the geometric relationships among core embeddings. The advantage of using the pairwise distribution is that it captures the relative geometric relationships among the embeddings as opposed to the absolute positions in the embedding space. For instance, in Figure 1, the embeddings invert after shaving the $k = 1$ shell, nevertheless, the relative distances among embeddings are largely unchanged. An alternative measure of stability would be the Frobenius norm of the difference in weighted adjacency matrices [8]; however, we found that the pairwise distribution provides a more granular measure of stability as opposed to a single norm value. After calculating the degenerate-core pairwise distribution at each k, `SHARE` measures the distance between the distributions with the Earth Mover's Distance (EMD).

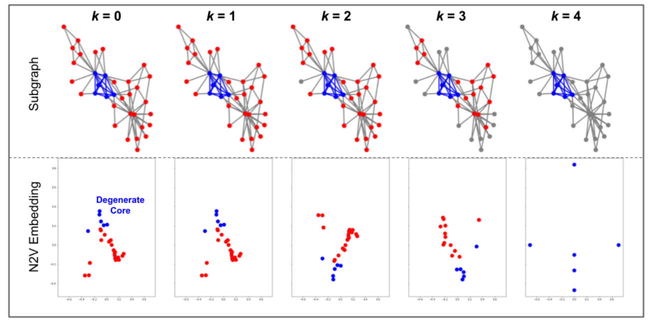


Figure 1: To measure the stability of degenerate-core embeddings, we use a method that we call SHave-And-Re-Embed, or "`SHARE`". In this figure, we apply `SHARE` to the Zachary Karate Club graph [30]. At each iteration, we shave off the outermost k-shell (top row) and re-embed the remaining subgraph (bottom row). We then analyze the stability of the degenerate-core embeddings. Removed nodes are shown in grey, remaining subgraph nodes in red, and degenerate-core nodes in blue. The Node2Vec embedding 2-D projection plots all use the same scale. The embeddings of the degenerate core vary widely as the periphery is removed.

In Equation 2.1, we define *instability* (Δ_k) at core k as the change in the DCPD after removing the previous k-shell, where D_k is the degenerate-core pairwise distribution for the k-core. Table 1 summarizes our notation.

$$\text{Instability } \Delta_k = \text{EMD}\left(D_k, D_{k-1}\right) \tag{2.1}$$

2.1 Graph Embedding Algorithms and Datasets

We ran the proposed stability analysis using a combination of graph embedding algorithms—namely, HOPE [20], Laplacian Eigenmaps [4], Node2Vec [9],

[1] Our code is available at https://github.com/dliu18/stable

[2] Our analysis pertains to undirected graphs without self-loops.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Symbol	Meaning
n, m	number of vertices, edges
$k_{\max}$	degeneracy of a graph
d	embedding dimension
D_k	the degenerate-core pairwise distribution (DCPD) for k-core k. See Sec. 2
Δ_k	instability at the k^{th} core. See Eqn. 2.1.
w_{ij}	weight of edge $\{i, j\}$
W_S	adjacency matrix for subgraph induced by a set of nodes S
$\mathcal{D}$	set of nodes in the degenerate core
$\boldsymbol{u}_i$	d-dimensional embedding for node i
$\boldsymbol{Y}$	$n \times d$ matrix where row i is $\boldsymbol{u}_i^T$
$\mathcal{L}_b, \mathcal{L}_s$	base and stability loss functions
α, β	regularization hyperparameters
n_b	number of training batches
η	learning rate

Table 1: Notations used in this paper.

SDNE [27], Hyperbolic GCN (HGCN) [6], and PCA as a baseline. We picked these graph embedding algorithms because they span the taxonomy provided in Chami et al. [5]. To find instability patterns, we experimented on a variety of graph datasets (listed in Table 2). The datasets are primarily from SNAP [14], with the exception of Wikipedia [16], Autonomous Systems (AS) [17], and the synthetic graphs, which were generated to be of similar size as the real-world graphs.

2.2 Perturbations and Downstream Performance To motivate the study of stability and graph embeddings we examine the effect of perturbations on downstream performance. Specifically, we conduct a link prediction evaluation where for a pair of nodes, the prediction score is the cosine similarity between the corresponding embeddings, and the ground-truth label is 1 if an edge exists and 0 otherwise. Figure 2 shows the link prediction AUC-ROC for each subsequent core of the Facebook network. When Laplacian Eigenmap embeddings are generated with the entire network of $n = 4039$ nodes, the embeddings achieve a link-prediction AUC-ROC of over 0.95. The performance remains strong until $k = 71$ when the AUC suddenly drops down to below 0.75. As we will show in Section 3.2.1, few characteristics distinguish the $k = 70$ and $k = 71$ cores. After the drop, performance recovers likely because the size of the cores is diminishing while the embedding dimension remains constant. Nonetheless, the performance drop at $k = 71$ motivates the need to understand the instability of graph embeddings.

Graph	n	m	$k_{\max}$	$\%\mathcal{D}$	$G_{\mathcal{D}}$ D
Wikipedia	4.8K	185K	49	3.1	0.526
Facebook	4.0K	88K	115	3.9	0.898
PPI	3.9K	77K	29	2.8	0.404
ca-HepTh	9.9K	26K	31	0.3	1.0
LastFM	7.6K	28K	20	0.6	0.614
AS	23K	48K	25	0.3	0.545
ER ($p = .002$)	5K	25K	7	67	0.002
ER ($p = .004$)	5K	50K	14	87	0.004
BA ($m = 5$)	5K	25K	5	100	0.002
BA ($m = 10$)	5K	50K	10	100	0.004
BTER (PA)	5K	25K	1	1.5	0.234
BTER (Arb.)	4.8K	35K	51	3.3	0.445

Table 2: Graph datasets used in our study. ER, BA, and BTER are short for Erdös-Rényi [7], Barabási-Albert [2], and Block Two-Level Erdös-Rényi [22] random graphs, respectively. PA is for a degree distribution that exhibits preferential attachment. Arb. is for an arbitrary degree distribution. A description of the synthetic graphs is in Appendix A.1. $\%\mathcal{D}$ is $|\mathcal{D}|/|G|$ and $G_{\mathcal{D}}$ D is the degenerate-core edge density.

3 Unstable Degenerate-Core Embeddings

We provide our results analyzing the stability of degenerate-core embeddings in three parts. First, we show that as k-shells are removed, the evolution of degenerate-core embeddings follows three patterns across various graph types and embedding algorithms. Second, we show that instability is correlated with increases in density. Third, we present a theorem showing that for Laplacian-Eigenmap embeddings of Erdös-Rényi graphs, the performance gap between the best and worst embeddings narrows with increased density.

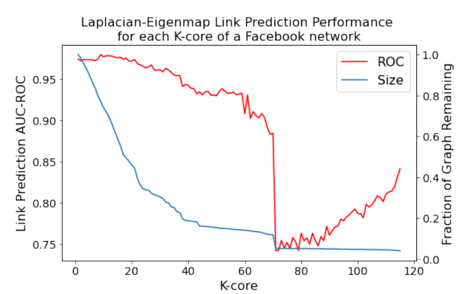


Figure 2: The link prediction performance for the Facebook network is sensitive to the removal of outer k-shells; notably the AUC-ROC drops by approximately 0.15 at $k = 71$. Performance rebounds after the dip likely because the number of nodes is decreasing while the embedding dimension ($d = 10$) remains constant.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

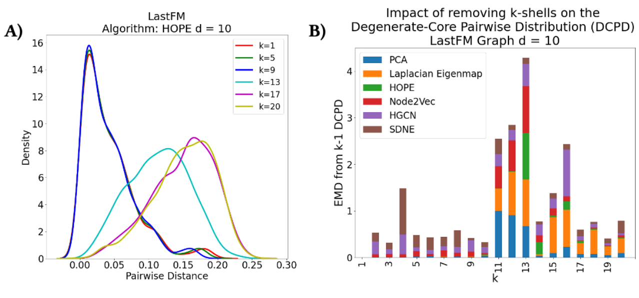
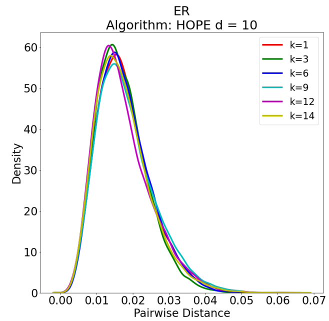
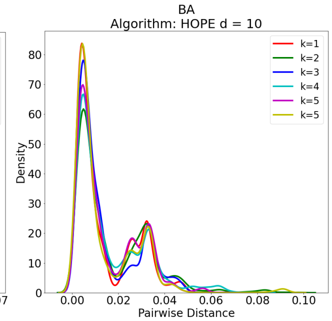


Figure 3: A) When we embed the LastFM graph with HOPE [20], we see that the degenerate-core pairwise distribution shifts abruptly between $k = 9$ and $k = 13$. This holds true for the various embedding dimensions that we tried. For brevity, we only show $d = 10$. B) We see that the pattern of abrupt shift generalizes across embedding algorithms for the LastFM graph. The y-axis is the instability as defined in Eq. 2.1 divided by the maximum instability value across all k. Across all algorithms we see high instability for $k \in \{11, 12, 13\}$.



(a) Erdös-Rényi



(b) Barabási-Albert

Figure 4: For Erdös-Rényi and Barabási-Albert graphs, the degenerate core embeddings were stable as the k-shells were removed iteratively. The degenerate-core pairwise distributions above are nearly identical regardless of the subgraph being embedded, and the pattern holds for all six embedding algorithms. This stability is likely due to the degenerate core constituting a large proportion of the entire graph.

3.1 Patterns in Stability of Degenerate-Core Structural Representation After running `SHARE` (our degenerate core stability method described in Section 2) on the 12 graphs and 6 embedding algorithms (described in Section 2.1), we observed the following three patterns.

Pattern 1: For many graph data, embedding algorithm combinations, the distribution of pairwise distances among degenerate core nodes shifts after removing specific k-shells, but is stable otherwise.

We observe that not only does the pairwise distribution change as k-shells are removed but the change often occurs at abrupt points. In Figure 3A, we show the degenerate-core pairwise distribution for the LastFM graph when embedded with HOPE. For readability, we plot the distribution at intervals of k. The distinguishing feature is that the distributions for $k = 1$ and $k = 9$ are quite similar (left-skewed). However, the distribution for $k = 13$ differs dramatically; then for $k > 13$, the distribution remains quite similar. To verify whether this pattern persists across graph embedding algorithms, Figure 3B shows the normalized EMD for each k-core across all algorithms tested, where we normalize by dividing by the largest EMD value per algorithm. We can see that across algorithms there is large instability in $k \in \{11, 12, 13\}$.

Pattern 2: The degenerate-core embeddings for the ER and BA graphs are stable.

In contrast to the real-world graphs, we found that the embeddings for the ER and BA graphs are stable. Figure 4 shows the degenerate-core pairwise distributions for the Erdös-Rényi and Barabási-Albert graphs. The distributions shown were generated with HOPE, however, the pattern holds for Node2Vec and Laplacian Eigenmaps as well. The stability for these random graphs is likely due to the fact that the degenerate core alone constitutes a large proportion of the entire graph as shown in Table 2, given the parameters that we selected. For this reason, removing the outer k-shells has less of an impact on the degenerate core.

Pattern 3: As k-shells are removed, the degenerate-core pairwise distribution becomes smoother and more unimodal.

We observe that not only does the degenerate-core pairwise distribution shift as k-shells are removed, but the distribution also loses modality. In Figure 5A, we analyze the HOPE embeddings for the Wikipedia graph. For larger k, the degenerate-core pairwise distribution is bimodal. When $k = 1$, the distribution has a small peak around the distance of 0.6. For $k > 40$, the distribution is nearly unimodal. We quantify this loss of modality with Hartigan's dip statistic [11], a value in $[0, 1]$ that measures the deviation from unimodality. In Figure 5, we see that across embedding algorithms for the Wikipedia graph, as k-shells are shaved, the dip statistic decreases in aggregate.

3.2 Significance of the Periphery In this section, we examine the causes of the patterns identified in the previous section. Using the Facebook and LastFM graphs as case studies, we see that embedding instability

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

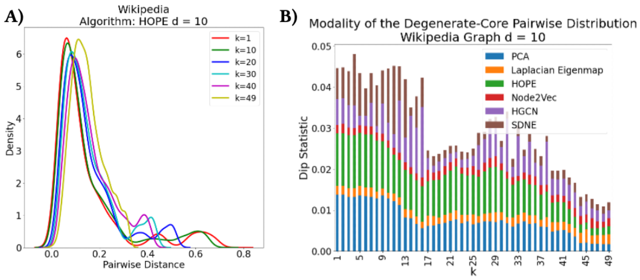


Figure 5: A) For the Wikipedia graph, as k-shells are removed, the degenerate-core pairwise distribution transforms from being bimodal to unimodal. We conjecture that the loss in modality is due to the loss in community structure. B) We quantify the loss of modality with Hartigan's dip statistic [11], where a value close to zero suggests unimodality. Aggregating across embedding algorithms, the statistic decreases as k-shells are shaved.

is correlated with subgraph edge density. We then generalize this result across all graphs by regressing instability on changes in subgraph features. Together, these findings show that the stability patterns found are not simply due to large numbers of nodes being removed from the graph but rather structural changes.

3.2.1 Case Studies: Facebook and LastFM Figure 6 shows the k-core embeddings before and after the point of maximum instability (the k with maximum Δ_k). For both Facebook and LastFM, before the maximum instability point, the degenerate core, colored in red, is separate from the periphery. In the case of Facebook, the 70-core exhibits core-periphery structure in which the degenerate core is the dense center. However, after one further k-shell removal, the degenerate-core embeddings become interspersed with the remaining subgraph. Figure 6 shows the results when embedding with Laplacian Eigenmaps for Facebook and with HOPE for LastFM. We observed that this pattern generalizes across various embedding dimensions and algorithms. The right-hand side of Figure 6 shows the subgraph (edge) density, size, and average clustering coefficient at each k-core. In particular, the point of greatest increase in edge density is denoted by the dashed red line. The figure shows that the point of greatest instability, shown on the left, corresponds more with an increase in density than a decrease in subgraph size.

3.2.2 Regression Analysis We model the relationship between changes in k-core subgraph features and the corresponding change in the degenerate-core pairwise distribution. The subgraph features we measure are the ratio between the number of nodes in the sub-

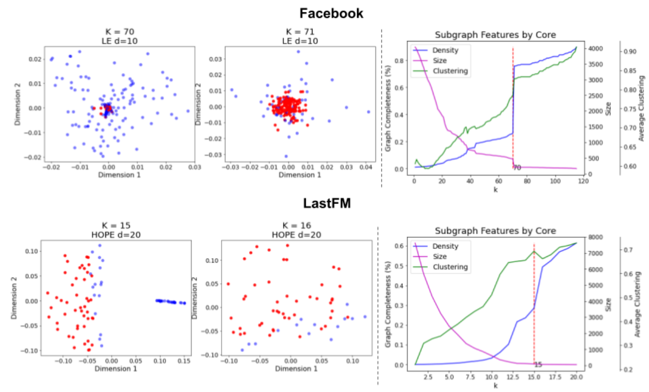


Figure 6: The point of maximum instability of embeddings for both the Facebook and LastFM graphs is correlated with increases in the subgraph density. The left-side scatter plots show the two-dimensional projection of the subgraph embeddings before and after the maximum instability point. In both plots, the degenerate core is colored red. The plots on the right-hand side track subgraph features with each k-shell removal. The point of highest edge density increase (dashed line) is also the maximum instability point.

graph and the number of nodes in G ("size"), edge density, average clustering coefficient, and transitivity, which are common features for characterizing subgraphs [1]. These features are inputs into the regression model shown in Equation 3.2, in which we correlate the change in the subgraph features with instability.

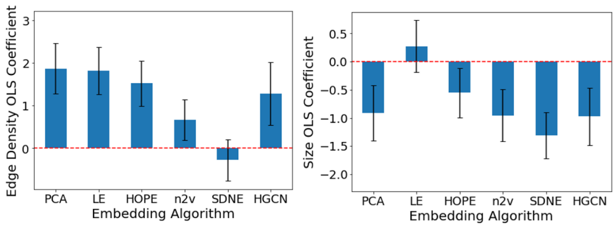
$$\begin{aligned}\Delta_k = \beta_0 &+ \beta_1 \Delta_{\text{size}} \\ &+ \beta_2\ \Delta_{\text{edge\_density}} \\ &+ \beta_3\ \Delta_{\text{clustering\_coefficient}} \\ &+ \beta_4\ \Delta_{\text{transitivity}}\end{aligned} \tag{3.2}$$

The data for the regression model was generated by examining consecutive k-cores. For the k and $k-1$ cores of a given graph, we measure the change in the aforementioned subgraph features as well as the instability, yielding one training data point. We repeat this process across all of the graphs to create a single dataset. Because the relationship between subgraph features and stability can vary by the embedding algorithm or dimension, we ran a regression model for each embedding algorithm and dimension combination.

Figure 7 shows the results of running the regression in Equation 3.2. The coefficients for edge density and size are grouped by embedding algorithm. We show the results for $d = 10$ for brevity. The error bars report 95% confidence intervals. Across all algorithms except SDNE, edge density is positively and statistically significantly correlated with instability. The negative co-

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy



(a) Edge Density (b) Size

Figure 7: The results of running the regression in Equation 3.2 show that increases in subgraph edge density are correlated with degenerate-core embedding instability. We ran a separate regression for every combination of embedding dimension and algorithm. The coefficients for edge density (a) and subgraph size (b) are shown above, where $d = 10$. For all algorithms except SDNE, the corresponding edge-density coefficient is positive and statistically significant. Reducing the graph size is also correlated with an increase in instability; however, for four of the six algorithms, the coefficient for size ($\Delta_{\text{size}} \in [-1, 0]$) is smaller in magnitude than the coefficient for density ($\Delta_{\text{edge\_density}} \in [0, 1]$). Error bars are provided for statistical significance ($p = 0.05$).

efficients for Δ_{size} indicate that a reduction in subgraph size is also correlated with an increase in instability; however, for four of the six algorithms, the coefficient for Δ_{size} is smaller in magnitude than the coefficient for $\Delta_{\text{edge\_density}}$, noting that both features are normalized. The coefficients for clustering coefficient and transitivity are predominantly statistically insignificant.

3.3 Theoretical Analysis of Erdös-Rényi Graphs Motivated by the empirical correlation between edge density and embedding instability, we theoretically analyze the quality of Laplacian Eigenmap embeddings for Erdös-Rényi graphs. Erdös-Rényi graphs are a natural model to consider because the edge probability parameter p is also the graph's expected edge density. In Theorem 3.1, we show that the performance gap between the best and worst Laplacian Eigenmap embeddings narrows as the expected edge density increases. The loss function for Laplacian Eigenmaps yields a lower loss for embeddings in which nodes close to each other in G are embedded closer to each other, as measured by Euclidean distance. As p, the expected edge density, increases, the range of loss values narrows, which implies that for large p, even the worst set of embeddings performs nearly as well as the best set.

Theorem 3.1. *Let G be an Erdös-Rényi graph with n nodes and edge probability p, and let $l_G(X)$ be the Laplacian Eigenmap loss for a set of embeddings $X \in \mathbb{R}^{n \times d}$ with embedding dimension d. Then, almost surely, the gap between the best set of embeddings and the worst set decreases as a function of p, where the ratio is lower-bounded as:*

$$\frac{min_X l_G(X)}{max_X l_G(X)} \geq \frac{\sqrt{(n-1)p} - o(1)}{\sqrt{(n-1)p} + o(1)}$$

Proof. See Appendix A.3 for the proof. □

4 STABLE: Algorithm for Stable Graph Embeddings

We propose a graph embedding algorithm `STABLE` that produces core-stable embeddings. `STABLE` augments any existing graph embedding algorithm with a differentiable objective function (the "base" objective) by adding an instability regularization. It is important to note that because `STABLE` optimizes an augmented objective function, the base loss for `STABLE` embeddings will be at least the base loss for the original, non-stable embeddings. Below, we outline our generic algorithm `STABLE` and show two instantiations of `STABLE` by augmenting Laplacian Eigenmaps and LINE. We use the notation introduced in Table 1.

4.1 Generic Algorithm

4.1.1 Objective function Our objective function consists of two components: the base objective $\mathcal{L}_b$ and an instability penalty $\mathcal{L}_s$. `STABLE` minimizes Equation 4.3 where α is a regularization hyperparameter.

$$\boldsymbol{Y}^* = \underset{\boldsymbol{Y} \in \mathbb{R}^{n \times d}}{\arg\min}\, \mathcal{L}_b(\boldsymbol{Y}, W) + \alpha \mathcal{L}_s(\boldsymbol{Y}, W, \mathcal{D}) \tag{4.3}$$

The instability penalty is high when the degenerate-core embedding is different in the following two cases: (1) the core is embedded in the context of the entire graph and (2) the core is embedded in isolation. We define $\hat{\boldsymbol{Y}}$ as the $|\mathcal{D}|$ x d matrix containing embeddings for the degenerate core when the core is isolated:

$$\hat{\boldsymbol{Y}}_D = \underset{\boldsymbol{Y} \in \mathbb{R}^{|D| \times d}}{\arg\min}\, \mathcal{L}_b(\boldsymbol{Y}, W_D) \tag{4.4}$$

Now, stability can be defined as the preservation of the first-order proximities between pairs of nodes in the degenerate core, where the first-order proximity p between embedding $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ is:

$$p(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{1}{1 + e^{-\boldsymbol{u}_i^T \boldsymbol{u}_j}} \tag{4.5}$$

We define the instability penalty as the sum of squares over all differences in first-order proximities between

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

pairs of degenerate-core nodes, where $\hat{\boldsymbol{u}}_i$ is the embedding of node i in $\hat{\boldsymbol{Y}}_D$:

$$\mathcal{L}_s = \sum_{i,j \in \mathcal{D}} \left| p\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right) - p\left(\hat{\boldsymbol{u}}_i, \hat{\boldsymbol{u}}_j\right) \right|^2 \tag{4.6}$$

To optimize `STABLE`'s objective function, we perform a batched stochastic gradient descent where each batch is a set of edges. For an edge $\{i, j\}$ where i and j are both in the degenerate core, the gradient for the instability penalty, with constants omitted, is as follows, where $z = \sigma\left(\boldsymbol{u}_i^T \boldsymbol{u}_j\right)$:

$$\frac{d\mathcal{L}_s}{d\boldsymbol{u}_i} = z\left[1 - z\right]\left[z - \sigma\left(\widehat{\boldsymbol{u}}_i^T \widehat{\boldsymbol{u}}_j\right)\right]\boldsymbol{u}_j \tag{4.7}$$

The expression for $d\mathcal{L}_s/d\boldsymbol{u}_j$ replaces the last term with $\boldsymbol{u}_i$. Further, the stability gradient is only used when both i and j are in the degenerate core. For edges outside of the degenerate core, the update rule only utilizes the gradient for the base objective.

Algorithm 1 `STABLE`

Require: G, W, n_b, α, η
Ensure: $\boldsymbol{Y}^*$
 $\mathcal{D} \leftarrow$ `degenerate_core`(G)
 $\widehat{\boldsymbol{Y}}_D \leftarrow$ `base_embed`(G_D, W_D)
 $\boldsymbol{Y} \leftarrow$ `base_embed`(G, W)
 $G \leftarrow$ `degenerate_clique`$(G, \mathcal{D})$
 $i \leftarrow 0$
 while $i < n_b$ **do**
  $i \leftarrow i + 1$
  edges $\leftarrow$ `sample_edges`(G)
  for i, j in edges **do**
   if $W[i, j] > 0$ **then**
    $\boldsymbol{Y}[i] \leftarrow \boldsymbol{Y}[i] - \eta\frac{d\mathcal{L}_b}{d\boldsymbol{u}_i}$ ▷ See Sec. 4.2
    $\boldsymbol{Y}[j] \leftarrow \boldsymbol{Y}[j] - \eta\frac{d\mathcal{L}_b}{d\boldsymbol{u}_j}$
   end if
   if $i \in \mathcal{D}$ and $j \in \mathcal{D}$ **then**
    $\boldsymbol{Y}[i] \leftarrow \boldsymbol{Y}[i] - \eta\alpha\frac{d\mathcal{L}_s}{d\boldsymbol{u}_i}$ ▷ See Eqn. 4.7
    $\boldsymbol{Y}[j] \leftarrow \boldsymbol{Y}[j] - \eta\alpha\frac{d\mathcal{L}_s}{d\boldsymbol{u}_j}$
   end if
  end for
 end while

Algorithm 1 provides pseudocode for `STABLE`. We begin by embedding the degenerate core in isolation ($\widehat{\boldsymbol{Y}}_D$) as well as the entire input graph (Y) to initialize the embeddings. Because the instability penalty $\mathcal{L}_s$ sums over all pairs of nodes in the degenerate core, not just connected nodes, the `degenerate_clique` method augments the graph by adding an edge between $\{i, j\} \forall i, j \in \mathcal{D}$. These edges are assigned weight zero and when drawn, only the stability update rule is applied and the base update rule is omitted.

4.2 Instantiations

4.2.1 Stable LINE When instantiated with LINE (first-proximity) [26], the base loss takes the form:

$$\mathcal{L}_b = -\sum_{i,j \in E} w_{ij} \log\left(p\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right)\right) \tag{4.8}$$

As is common with LINE implementations, for computational efficiency we utilize negative sampling such that for a sampled edge i, j we minimize the following:

$$-\log\left(p\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right)\right) - E_{j' \sim P_n}\left[\log\left(p\left(-\boldsymbol{u}_i, \boldsymbol{u}_{j'}\right)\right)\right] \tag{4.9}$$

The gradient for each vertex $\boldsymbol{u}_i, \boldsymbol{u}_j, \boldsymbol{u}_{j'}$ is:

$$\begin{aligned} \frac{d\mathcal{L}_b}{d\boldsymbol{u}_i} &= -(1 - \sigma(\boldsymbol{u}_i^T \boldsymbol{u}_j))u_j + \sum_{j'} \sigma(\boldsymbol{u}_i^T \boldsymbol{u}_{j'}))\boldsymbol{u}_{j'} \\ \frac{d\mathcal{L}_b}{d\boldsymbol{u}_j} &= -(1 - \sigma(\boldsymbol{u}_i^T \boldsymbol{u}_j))u_i \\ \frac{d\mathcal{L}_b}{d\boldsymbol{u}_{j'}} &= \sigma(\boldsymbol{u}_i^T \boldsymbol{u}_{j'}))\boldsymbol{u}_i \end{aligned} \tag{4.10}$$

The complexity when instantiated with LINE is $\mathcal{O}\left(ndb + m\right)$ where b is the number of negative samples per edge; d is the number of embedding dimensions; and n, m are the number of nodes and edges, respectively. The first term accounts for computing b gradients of size d for n samples and the second term accounts for the overhead needed to set up the edge sampling data structures. Of note, this is the same complexity as LINE itself, so `STABLE` does not add to the runtime complexity.

4.2.2 Stable Laplacian Eigenmaps Laplacian Eigenmaps [4] optimizes the following objective function f [4]:

$$f\left(W, \boldsymbol{Y}\right) = \sum_{i,j} w_{ij} \|\vec{u_i} - \vec{u_j}\|^2 \tag{4.11}$$

However, the optimization is performed over the feasible set $Y^T D Y = I$, where D is the diagonal matrix such that D_{ii} is the degree of node i. `STABLE` initializes with Laplacian Eigenmaps embeddings. For this reason, instead of performing a constrained optimization, we penalize embeddings that deviate from the initial values; adding a deviation penalty proportional to the norm of the difference from the initial embeddings Y_0. To balance the orders of magnitude for these two losses, we introduce a hyperparameter β. Thus, the base objective when instantiated with Laplacian Eigenmaps is:

$$\mathcal{L}_b = f\left(W, \boldsymbol{Y}\right) + \beta \left\|Y - Y_0\right\|^2 \tag{4.12}$$

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Where the gradient is:

$$\frac{d\mathcal{L}_b}{d\boldsymbol{u}_i} = w_{ij}\left(\boldsymbol{u}_i - \boldsymbol{u}_j\right) + \beta\left(\boldsymbol{u}_i - \boldsymbol{u}_{i0}\right) \tag{4.13}$$

The runtime complexity for the Laplacian Eigenmap instantiation is $\mathcal{O}(nd + m)$ because negative sampling is not used.

5 Experiments

Our experiments show that `STABLE` produces embeddings that are both core-stable and accurate for link prediction, using the methodology described in Section 2.2. We have included the details of our experimental setup in Appendix A.5. Table 3 lists the results from our link prediction experiments. For each graph and algorithm configuration, we ran five trials using random edge sets and report the mean and confidence interval. `STABLE`'s graph embeddings preserve and at times improve upon the link-prediction accuracy of the original embeddings. For Laplacian Eigenmaps, relative to the base AUCs, the `STABLE` AUCs are lower by 0.031 on average across the six real-world graphs. For the Autonomous Systems and Wikipedia graphs, `STABLE` yields marginally higher AUCs. With LINE, `STABLE` preserves performance even more closely as the `STABLE` AUCs are lower by only 0.003 on average. In Appendix A.6 we verify that preserving performance occurs in tandem with decreasing the stability penalty $\mathcal{L}_s$.

6 Related Work

We review related work on k-core analysis and the limitations of graph embeddings. **k-core Analysis.** k-core structure has been important for understanding spreading processes on graphs, in particular identifying the most-influential spreaders [19]. Common patterns related to k-core structure have been identified such as correlations between a node's degree and coreness (largest k such that the node is in the k-core) as well as community structure in dense cores [24]. Recent work has also broadened the study of k-cores to consider the addition of "anchor nodes" that prevent large cascades when individual core nodes are removed [13]. Finally, it has been shown that not all degenerate cores are equally important; the most salient degenerate cores, called "true cores", are those that are well interconnected with outer shells [15]. **Limitations of Graph Embeddings.** Practitioners are tasked with choosing from a large selection of algorithms [31] and even once an algorithm has been chosen, hyperparameters such as the embedding dimension can greatly affect performance [10]. Further, in the case of community detection, expensive algorithms do not always perform traditional algorithms [25]. Recent work has also established more theoretical limits to graph embeddings, showing that at low dimensions it is impossible for graph embeddings to capture the triangle richness of real-world networks. Stability has also been identified as an issue [21], however, this study defined stability in a different sense: the consistency of embeddings when the algorithm is re-run multiple times.

7 Conclusion

The degenerate core of a graph is seen as the most influential part of that graph. In this work, we examined the stability of embeddings for the nodes in the degenerate core. We defined stability as the property of being resilient to perturbations. We defined perturbations as removing k-shells iteratively from the graph. We observed three patterns of instability across a variety of popular graph embedding algorithms and numerous real-world and synthetic data sets. We also correlated abrupt points of instability with increases in edge density. Subsequently, we introduced `STABLE`: an algorithm that takes an existing graph embedding algorithm and adds a stability objective. We showed how `STABLE` works on two popular graph embedding algorithms and reported experiments that showed the value of `STABLE`.

References

[1] B. Abrahao, S. Soundarajan, J. Hopcroft, and R. Kleinberg, *On the separability of structural classes of communities*, in KDD, 08 2012, pp. 624–632.

[2] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.

[3] P. Barberá, N. Wang, R. Bonneau, J. T. Jost, J. Nagler, J. Tucker, and S. González-Bailón, *The critical periphery in the growth of social protests*, PLOS ONE, 10 (2015), pp. 1–15.

[4] M. Belkin and P. Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Computation, 15 (2003), pp. 1373–1396.

[5] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, *Machine learning on graphs: A model and comprehensive taxonomy*, 2021, https://arxiv.org/abs/2005.03675v2.

[6] I. Chami, Z. Ying, C. Ré, and J. Leskovec, *Hyperbolic graph convolutional neural networks*, in NeurIPS, 2019, pp. 4869–4880.

[7] P. Erdös and A. Rényi, *On the evolution of random graphs*, Publ. Math. Inst. Hung. Acad. Sci., 5 (1960), p. 17.

[8] P. Goyal, N. Kamra, X. He, and Y. Liu, *Dyngem: Deep embedding method for dynamic graphs*, 2018, https://arxiv.org/abs/1805.11273.

[9] A. Grover and J. Leskovec, *Node2vec: Scalable feature learning for networks*, in KDD, 2016, p. 855–864.

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/19/23 to 208.30.97.64 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Table 3: Area under the ROC (a.k.a. AUC) performance on link prediction. Base refers to the original embeddings of Laplacian Eigenmaps and LINE. `STABLE` refers to the embeddings produced by `STABLE` when instantiated with Laplacian Eigenmaps and LINE. For each graph and algorithm configuration, we ran five trials with randomly selected edge sets and report the mean and confidence interval rounded to three decimal places. On average, `STABLE` and Base have similar AUC values on link prediction. We highlight the cases in which the performance gap between Base and `STABLE` is statistically significant ($p \leq 0.05$) as determined by unpaired t-tests.

	Laplacian Eigenmaps AUC		**LINE AUC**	
Graph	Base	`STABLE`	Base	`STABLE`
Facebook	**0.982 ± 0.000**	**0.924 ± 0.047**	**0.971 ± 0.001**	**0.933 ± 0.001**
LastFM	**0.910 ± 0.001**	**0.785 ± 0.001**	**0.914 ± 0.001**	**0.895 ± 0.002**
ca-HepTh	0.811 ± 0.008	0.811 ± 0.008	**0.893 ± 0.003**	**0.890 ± 0.001**
Protein-Protein	0.770 ± 0.016	0.761 ± 0.008	0.638 ± 0.036	0.660 ± 0.022
Autonomous Systems	0.693 ± 0.002	0.699 ± 0.020	**0.693 ± 0.007**	**0.672 ± 0.004**
Wikipedia	0.614 ± 0.001	0.615 ± 0.001	**0.458 ± 0.008**	**0.499 ± 0.003**

[10] W. Gu, A. Tandon, Y.-Y. Ahn, and F. Radicchi, *Principled approach to the selection of the embedding dimension of networks*, Nature Communications, 12 (2021), p. 3772.

[11] P. M. Hartigan, *Algorithm as 217: Computation of the dip statistic to test for unimodality*, Journal of the Royal Statistical Society. Series C (Applied Statistics), 34 (1985), pp. 320–325.

[12] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, *Representation learning for dynamic graphs: A survey*, JMLR, 21 (2020), pp. 1–73.

[13] R. Laishram, A. E. Sar, T. Eliassi-Rad, A. Pinar, and S. Soundarajan, *Residual Core Maximization: An Efficient Algorithm for Maximizing the Size of the k-Core*, 2020, pp. 325–333.

[14] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*. `http://snap.stanford.edu/data`, June 2014.

[15] Y. Liu, M. Tang, T. Zhou, and Younghae Do, *Core-like groups result in invalidation of identifying super-spreader by k-shell decomposition*, Scientific Reports, 5 (2015), p. 9602.

[16] M. Mahoney, *Large text compression benchmark*, 2011, `http://www.mattmahoney.net/dc/textdata`.

[17] M. Newman, *Internet network data*, 2006, `http://www-personal.umich.edu/~mejn/netdata/`.

[18] M. E. J. Newman, *Network structure from rich but noisy data*, Nature Physics, 14 (2018), pp. 542–545.

[19] S. Osat, F. Radicchi, and F. Papadopoulos, *k -core structure of real multiplex networks*, Physical Review Research, 2 (2020).

[20] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, *Asymmetric transitivity preserving graph embedding*, in KDD, 2016, p. 1105–1114.

[21] T. Schumacher, H. Wolf, M. Ritzert, F. Lemmerich, J. Bachmann, F. Frantzen, M. Klabunde, M. Grohe, and M. Strohmaier, *The effects of randomness on the stability of node embeddings*, 2020, `https://arxiv.org/abs/2005.10039`.

[22] C. Seshadhri, T. G. Kolda, and A. Pinar, *Community structure and scale-free collections of erdös-rényi graphs*, Phys. Rev. E, 85 (2012), p. 056109.

[23] C. Seshadhri, A. Sharma, A. Stolman, and A. Goel, *The impossibility of low-rank representations for triangle-rich complex networks*, PNAS, 117 (2020), pp. 5631–5637.

[24] K. Shin, T. Eliassi-Rad, and C. Faloutsos, *Corescope: Graph mining using k-core analysis - patterns, anomalies and algorithms*, in ICDM, 2016, pp. 469–478.

[25] A. Tandon, A. Albeshri, V. Thayananthan, W. Alhalabi, F. Radicchi, and S. Fortunato, *Community detection in networks using graph embeddings*, Phys. Rev. E, 103 (2021), p. 022316.

[26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, *Line: Large-scale information network embedding*, in WWW, 2015, p. 1067–1077.

[27] D. Wang, P. Cui, and W. Zhu, *Structural deep network embedding*, in KDD, 2016, p. 1225–1234.

[28] K. Wiesner, A. Birdi, T. Eliassi-Rad, H. Farrell, D. Garcia, S. Lewandowsky, P. Palacios, D. Ross, D. Sornette, and K. Thébault, *Stability of democracies: a complex systems perspective*, European Journal of Physics, 40 (2018), p. 014002.

[29] J.-G. Young, G. T. Cantwell, and M. E. J. Newman, *Bayesian inference of network structure from unreliable data*, Journal of Complex Networks, 8 (2021).

[30] W. W. Zachary, *An information flow model for conflict and fission in small groups*, J. of Anthropological Research, 33 (1977), pp. 452–473.

[31] Y.-J. Zhang, K.-C. Yang, and F. Radicchi, *Systematic comparison of graph embedding methods in practical tasks*, Physical Review E, 104 (2021), p. 044315.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited