Tracking the Adjacency Spectral Embedding for Streaming Graphs

Federico Larroca*, Paola Bermolen*‡, Marcelo Fiori*‡, Bernardo Marenco*‡, and Gonzalo Mateos†
*Facultad de Ingeniería, Universidad de la República, Uruguay
Email: {flarroca,paola,mfiori,bmarenco}@fing.edu.uy
‡ Centro Interdisciplinario en Ciencia de Datos y Aprendizaje Automático (CICADA), Universidad de la República, Uruguay
†Dept. of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA.
Email: gmateosb@ur.rochester.edu

***Abstract*—The popular Random Dot Product Graph (RDPG) generative model postulates that each node has an associated (latent) vector, and the probability of existence of an edge between two nodes is their inner-product (with variants to consider directed and weighted graphs). In any case, the latent vectors may be estimated through a spectral decomposition of the adjacency matrix, the so-called Adjacency Spectral Embedding (ASE). Assume we are monitoring a stream of graphs and the objective is to track the latent vectors. Examples include recommender systems or monitoring of a wireless network. It is clear that performing the ASE of each graph separately may result in a prohibitive computation load. Furthermore, the invariance to rotations of the inner product complicates comparing the latent vectors at different time-steps. By considering the minimization problem underlying ASE, we develop an iterative algorithm that updates the latent vectors' estimation as new graphs from the stream arrive. Differently to other proposals, our method does not accumulate errors and thus does not requires periodically re-computing the spectral decomposition. Furthermore, the pragmatic setting where nodes leave or join the graph (e.g. a new product in the recommender system) can be accommodated as well. Our code is available at https://github.com/marfiori/efficient-ASE.**

***Index Terms*—graph representation learning, node embeddings, graph sequence.**

I. Introduction

There is an increasing interest in statistical analysis of data stemming from networks that are observed sequentially [1]–[3]. These may come from a social network (new users sign up to the platform and followers are gained and lost over time), a recommender system (new content is added and user's preferences are revealed over time) or a transportation network (where the edges' weight indicate the traffic intensity of a route between two nodes at a given time).

One of the most popular statistical models to analyze a single of such graphs $G = (V, E)$ is the so-called Random Dot Product Graph (RDPG) [4]. This model postulates that each node is associated to a latent vector $\mathbf{x}_i \in \mathbb{R}^d$ (which is unobserved and may be interpreted as an embedding) and the probability of existence of an edge between nodes i and j is the inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$. The case of directed and/or weighted graphs are easily accounted for through relatively simple variations of the base model. Including as particular cases other generative models (such as Stochastic Block Models), its great expressiveness and the interpretability of the resulting embeddings are the reasons behind its popularity.

Assume that we are observing a stream of graphs G_t and the objective is to track the underlying embedding for all time-steps t. A naive approach would be to compute separately for each t the estimated embeddings. As we discuss in some detail in the next section, these estimations are obtained by computing the d most significant eigenvalues of the adjacency matrix $\mathbf{A}_t$ and their associated eigenvectors, a procedure known as Adjacency Spectral Embedding (ASE). As the graph size increases, computing the ASE for each graph G_t separately will result in an extremely computationally expensive algorithm.

Furthermore, since it is defined in terms of the inner product, it is easy to see that RDPG is invariant to rotations of the embeddings. Thus, computing the ASE for each graph G_t separately may result in significantly different embeddings for each t (i.e. rotated versions of the same embeddings), while actually nothing changed on the underlying generative process. Note that we may try to find the best alignment between the previous and the current embedding (solving a so-called Procrustes problem), but this approach will only worsen the computational cost of the algorithm.

Proposed approach and contributions. Instead of computing the eigenpairs, we revisit the underlying optimization problem. Notice that the ASE is basically looking for the best d-rank approximation of $\mathbf{A}_t$ in terms of the Frobenius norm. This interpretation along with recent advances in first-order non-convex optimization allows us to address the problem through Gradient Descent (GD). In particular, our proposal is to compute the ASE of graph G_t by running a gradient descent initialized on the embeddings of the previous time-step (i.e. the ASE of G_{t-1}). This warm started gradient descent method maintains a certain alignment between the latent positions, while at the same time converging in few iterations.

Furthermore, we also extend the method to the case where new nodes appear in the network. This so-called inductive learning is generally regarded as beyond the capabilities of shallow embeddings as the one we are discussing here [5], [6,

This work was partially funded by the NSF (awards CCF-1750428 and ECCS-1809356).

ch. 3.4]. However, a simple approximation through projection may be used as the initialization of the new nodes in the gradient descent method, thus enabling inductive learning in the context of ASE without requiring the re-computation of the eigenpairs.

Related work. The rotation ambiguity of RDPG has long been recognized as a problem when estimating the embeddings for several graphs. To address this issue, the Omnibus Embedding [7] or the Unfolded Adjacency Spectral Embedding (UASE) [8], [9] propose to compute the ASE of different forms of concatentations of the adjacency matrices $\mathbf{A}_t$. For instance, when estimating the embeddings of T matrices, the Omnibus embedding requires computing the eigenpairs of a matrix of size $Tn \times Tn$ (with n the number of nodes in the graph), whereas UASE requires computing the Singular Value Decomposition of an $n \times Tn$ one.

In addition to being enormously costly as T grows beyond very small values, these techniques are designed for the batch case, where we are able to observe the whole sequence of graphs G_t. In the streaming context as the one we consider here, and since ASE may be regarded as computing the eigenpairs, we may instead resort to classic methods that recursively update these values as the adjacency matrix is modified [10]. However, these algorithms are computationally expensive except for specific types of changes (e.g. rank-1 modifications) and, as it has been pointed out in the literature and we illustrate in our simulations, they accumulate error as t increases [11] and are also susceptible to the rotation problem.

Finally, the problem of computing the ASE as new nodes are introduced in the network has been studied in the context of stringent memory requirements, where only the embeddings are kept and a single new node is added [12]. In this case, projecting the new column (of the adjacency matrix) to the space spanned by the older embeddings produces asymptotically consistent estimates of the new node's embedding. However, the underlying assumption is that the older embeddings do not change over time, and as several nodes are added to the network the error tends to accumulate too. We will nevertheless use this estimate as the warm start of the new nodes for our gradient descent method, thus improving on the resulting precision as several nodes are added, while at the same time permitting changes of the older nodes' embedding.

II. Preliminaries and problem statement

A. Random dot product graphs

Consider an unweighted and undirected graph $G = (V, E)$, with nodes $V = \{1, \dots, n\}$ and edges $E \subseteq V \times V$. In this section, we will drop the time-step sub-index t for the sake of clarity. In the RDPG model each node $i \in V$ has an associated column vector $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$, and edge (i, j) exists with probability $\mathbf{x}_i^\top \mathbf{x}_j$ (a particular case of the latent space model [13]). Note that the set $\mathcal{X}$ of possible $\mathbf{x}_i$ is such that $\mathbf{x}^\top \mathbf{y} \in [0, 1]$, $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$. In general, vectors $\mathbf{x}_i$ may be random, drawn from a distribution in $\mathcal{X}$.

Thus, letting $\mathbf{A} \in \{0,1\}^{n \times n}$ be the random symmetric adjacency matrix of G and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ the matrix of latent vertex positions, the RDPG model specifies

$$\mathrm{P}\left(\mathbf{A} \,\middle|\, \mathbf{X}\right) = \prod_{i<j} (\mathbf{x}_i^\top \mathbf{x}_j)^{A_{ij}} (1 - \mathbf{x}_i^\top \mathbf{x}_j)^{1 - A_{ij}}. \qquad (1)$$

That is, *given* $\mathbf{X}$, edges are conditionally independent with $A_{ij} \sim \mathrm{Ber}(\mathbf{x}_i^\top \mathbf{x}_j)$. The RDPG model is a tractable yet expressive family of random graphs that subsume Erdös-Rényi (ER) and SBM ensembles as particular cases. Indeed, if $\mathbf{x}_i = \sqrt{p}\ \forall i$, we obtain an ER graph with edge probability p. An SBM with M communities may be generated by restricting $\mathbf{X}$ to having only (at most) M different columns (i.e. $|\mathcal{X}| = M$); see also [4] for additional examples. Furthermore, the weighted case is easily accommodated by the model. In this case, the inner-product between nodes' embeddings is equal to the mean of the weight's distribution [14].

B. Inference on RDPG

Given a graph stemming from an RDPG with adjacency matrix $\mathbf{A}$, we now discuss how to estimate the matrix $\mathbf{X}$ of latent vertex positions. In lieu of a maximum-likelihood estimator that is intractable beyond toy graphs, the key intuition is that $\mathbf{A}$ is a noisy observation of

$$\mathbf{P} = \mathbf{X}\mathbf{X}^\top, \qquad (2)$$

the matrix of edge probabilities p_{ij}, since $\mathbb{E}\left[\mathbf{A} \,\middle|\, \mathbf{X}\right] = \mathbf{P}$. Therefore, and remembering that the diagonal entries of $\mathbf{P}$ are zero, we want to solve the following problem [15]:

$$\hat{\mathbf{X}} \in \underset{\mathbf{X} \in \mathbb{R}^{N \times d}}{\operatorname{argmin}} \|\mathbf{M} \circ (\mathbf{A} - \mathbf{X}\mathbf{X}^\top)\|_F^2, \qquad (3)$$

Instead, the usual approach to obtain an approximate solution of (3) is to slightly modify the problem in order to avoid the zero-diagonal constraint (either by replacing the main diagonal of $\mathbf{A}$, or simply ignoring the constraint) [4]:

$$\hat{\mathbf{X}} \in \underset{\mathbf{X}}{\operatorname{argmin}} \|\mathbf{A} - \mathbf{X}\mathbf{X}^\top\|_F^2, \text{ s. to } \operatorname{rank}(\mathbf{X}) = d. \qquad (4)$$

The solution to (4) is readily given by

$$\hat{\mathbf{X}} = \hat{\mathbf{Q}}\hat{\mathbf{\Lambda}}^{1/2}, \qquad (5)$$

where $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$ is the spectral decomposition of $\mathbf{A}$, $\hat{\mathbf{\Lambda}} \in \mathbb{R}^{d \times d}$ is a diagonal matrix with the d largest eigenvalues of $\mathbf{A}$, and $\hat{\mathbf{Q}} \in \mathbb{R}^{n \times d}$ are the corresponding d dominant eigenvectors. We are assuming that $\hat{\mathbf{\Lambda}}$ has only non-negative values, a limitation that may be easily circumvented [16]. Estimator (5) is known as the Adjacency Spectral Embedding (ASE).

C. Problem statement and proposed solution

Assume we observe a stream of graphs G_t and the objective is to estimate its embedding $\hat{\mathbf{X}}_t$. Note that both the number of edges as well as nodes may change between time-steps, although we assume we can identify nodes at all time-steps (e.g. they correspond to users in a social network or nodes in a wireless network).

In order to compute the new ASE $\hat{\mathbf{X}}_t$, we have the estimate of the previous time-step $\hat{\mathbf{X}}_{t-1}$, which we would like to compare to the new one (for instance, in the context of change-point detection [14], visualization or any other downstream task). As we discussed before, computing the eigenpairs for each time-step separately is out of the question due to its computational cost, as well as the potentially rotated estimates. An alternative is to update these eigenpairs through the difference in the adjacency matrix $\mathbf{\Delta}_t = \mathbf{A}_t - \mathbf{A}_{t-1}$ [10]. However, as we will demonstrate through simulations in the next section, this solution also suffers from the rotation ambiguity problem, as well as accumulating errors as t grows.

Following the ideas first presented in [17], we propose instead to reconsider the original optimization problem (3) and solve it through GD instead of computing the eigenpairs. Let us denote by $\hat{\mathbf{X}}_t[k]$ the k-th iteration of the algorithm in time-step t and by $f : \mathbb{R}^{n\times d} \to \mathbb{R}$ the objective function $f(\mathbf{X}) = \|\mathbf{M}_t \circ (\mathbf{A}_t - \mathbf{X}\mathbf{X}^\top)\|_F^2$. The GD algorithm is then

$$\hat{\mathbf{X}}_t[k+1] = \hat{\mathbf{X}}_t[k] - \alpha \nabla f(\hat{\mathbf{X}}_t[k]), \quad k = 0, 1, 2, \dots \tag{6}$$

where $\alpha > 0$ is the step size and $\nabla f(\mathbf{X}) = 4\left[\mathbf{M}_t \circ (\mathbf{X}\mathbf{X}^\top - \mathbf{A}_t)\right]\mathbf{X}$, for symmetric $\mathbf{A}_t$ and $\mathbf{M}_t$.

Note that the objective function depends on the product $\mathbf{X}\mathbf{X}^\top$, and is convex when considering the variable $\mathbf{Z} = \mathbf{X}\mathbf{X}^\top$. In this context, this approach is sometimes called *factorized GD* [18], or *Procrustes flow* [19], which has been shown to converge with linear rate to the solution $\hat{\mathbf{X}}_t$ of (3) if the initial value $\hat{\mathbf{X}}_t[0]$ is close [18], [20].

Precisely, our idea is to initialize the GD by using $\hat{\mathbf{X}}_t[0] = \hat{\mathbf{X}}_{t-1}$; i.e. the estimate of the previous time-step. Given the above results, as long as the underlying embeddings do not change dramatically between time-steps, the GD will converge in few iterations to the new solution. Even if the embeddings do change significantly at time-step t, the algorithm is guaranteed to converge except for some very specific initializations which correspond to stationary points. In our experience this pathological case has never occurred.

The question remains on how to proceed when nodes leave or join the network. The former is straightforward, since we may simply keep the rows of $\hat{\mathbf{X}}_{t-1}$ corresponding to the nodes that remain in the network at time-step t and use that as an initialization of the GD. The latter is more challenging.

Assume for clarity of the exposition that a single new node $i = n+1$ is added to the network at time-step t. The new adjacency matrix $\mathbf{A}_t$ now has an extra row (column) $\mathbf{a}_{n+1} \in \{0,1\}^n$ (where the $(n+1)$-th coordinate, corresponding to the diagonal of $\mathbf{A}_t$, will be zero). Instead of randomly initializing the new embedding $\hat{\mathbf{x}}_{n+1}$, we project $\mathbf{a}_{n+1}$ to the d-dimensional space spanned by the columns of $\hat{\mathbf{X}}_{t-1}$. That is to say:

$$(\hat{\mathbf{X}}_t[0])_{n+1} = \mathbf{a}_{n+1}\hat{\mathbf{X}}_{t-1}^{\text{norm}}, \tag{7}$$

where $\hat{\mathbf{X}}_{t-1}^{\text{norm}}$ is the column-wise normalized version of $\hat{\mathbf{X}}_{t-1}$. For the rest of the entries of $\hat{\mathbf{X}}_t[0]$ we use $\hat{\mathbf{X}}_{t-1}$ as before.

This initialization has the advantage of being very simple and it is actually a consistent estimator of the true new $\mathbf{x}_{n+1}$ as $n \to \infty$ [12]. Note however that if $m > 1$ nodes are added to the network at time-step t, this initialization alone will forcefully ignore the information contained in the links between the new nodes. Furthermore, even if nodes are added one at a time, the error introduced in the estimation of the first $\hat{\mathbf{X}}_0$ will negatively impact the estimation error of the successive $\hat{\mathbf{X}}_t$. In a nutshell, running GD initialized at (7) will both include information of the new nodes to correct the old estimates $\mathbf{X}_{t-1}$ as well as the interconnection between these new nodes. This will be further illustrated in the next section.

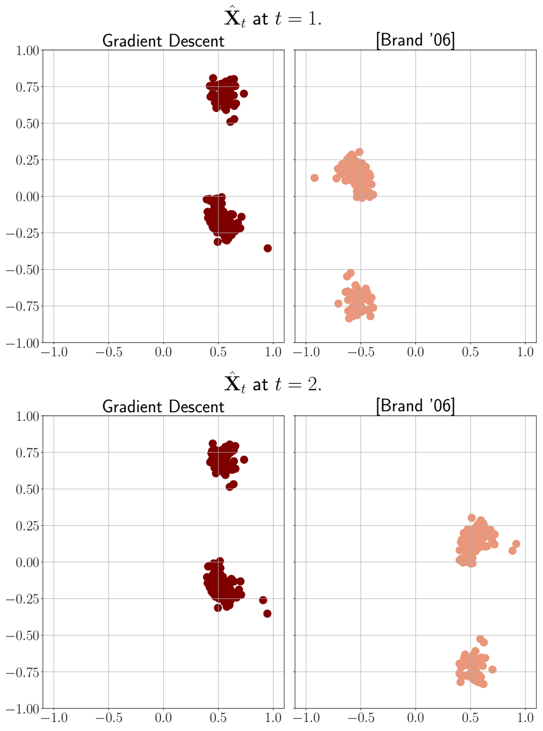

Fig. 1. The embeddings corresponding to the first two time-steps of our method (left) and by updating the corresponding eigenpairs through a classic iterative method [10] (right) when in a two-class SBM a single node changes affiliation at each t. Note how the change of a single node produces significantly different results for [10], whereas our method provides stable results.

III. Numerical tests

A. Simulated data

Embeddings' stability. Let us first consider a relatively simple case, where the number of nodes is fixed but the underlying embeddings change. In particular, we will consider an SBM graph with $n = 200$ nodes and two communities. At each time-step t a single randomly chosen node changes affiliation to the other community. The resulting embeddings of our method for the first two time-steps are displayed on the left of Fig. 1. See how effectively the embedding of a single node is moved from one community to the other, while the rest of the embeddings remain virtually unmoved.

Furthermore, the figure also displays on the right the result of applying the method described in [10] to re-compute the ASE at the same time steps. This is a classic iterative

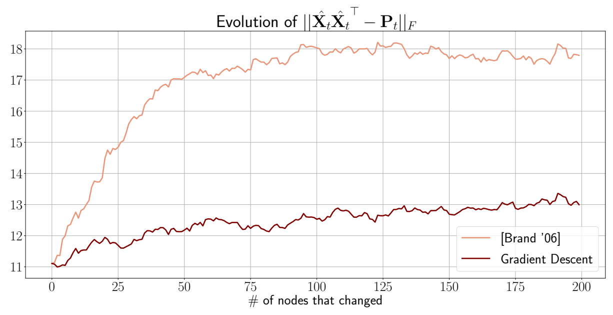

Fig. 2. The error in estimating the probability matrix using our method and the one described in [10] when in a two-class SBM a single node changes affiliation at each t. Note how our method produces approximately the same error, whereas the method in [10] accumulates error as t progresses.

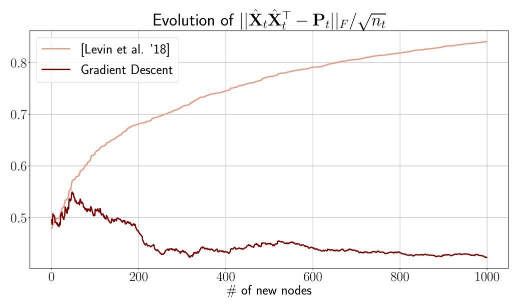

Fig. 3. The error in estimating the probability matrix using our method and the one described in [12] for the case when at each t a single new node is added to an Erdös-Rényi graph. Note again how our method produces an estimate with a constant normalized error.

method that updates the d most significant eigenpairs based on $\mathbf{\Delta}_t = \mathbf{A}_t - \mathbf{A}_{t-1}$. Note how, similarly to re-computing the ASE at each time-step, a single node may produce significantly different embeddings (i.e. rotations). Furthermore, and in addition to these rotations, since this method updates only the d most significant eigenpairs, the error in terms of $\|\hat{\mathbf{X}}_t\hat{\mathbf{X}}_t^T - \mathbf{P}_t\|_F$ also increases with t. This is shown in figure 2, and it is a known drawback of this kind of methods [11]. On the other hand, and as illustrated in the same figure, our method produces estimates with a constant error as expected.

Nodes' addition. An apparent drawback of analyzing a stream of graphs using a spectral decomposition such as ASE is its inability to deal with the inductive setting. As new nodes join the network, it is indeed not clear at all how to compute their embeddings, except by performing a complete re-computation of all the nodes.

However, this is not the only possibility, and as we discussed before we may obtain a first approximation of these embeddings by projecting the new rows of the adjacency matrix to the space spanned by the current embeddings (c.f. equation (7)) [12]. But using this approximation alone has at least three drawbacks. First of all, the new nodes are not used to improve on the estimate of the older nodes' embeddings (e.g. in an SBM, if the new nodes belong to the same community as some of the older ones, the resulting estimation should have a smaller variance [4]). Secondly, the interconnection between the new nodes is disregarded in this estimation. And lastly, the implicit assumption is that the older nodes' embedding do not change.

Our proposal is to initialize the embeddings of the nodes added at time-step t using the projection, and then perform GD on all of the embeddings as usual. We now illustrate through a simple example how this results in a method that circumvents the two first issues mentioned before (the last one was illustrated in the last subsection). Let us consider an Erdös-Rényi graph with a fixed connection probability $p = 0.1$ and with an initial number of nodes $n_0 = 100$. At each time-step t we add a single node under the same model, so that $n_t = n_{t-1} + 1$. The normalized error in terms of $\|\hat{\mathbf{X}}_t\hat{\mathbf{X}}_t^T - \mathbf{P}_t\|_F/\sqrt{n_t}$ is displayed in figure 3. Note how our method obtains a constant normalized error as expected, whereas simply projecting accumulates error. It is important to highlight that this increase in precision is gained with a modest increase in computation (some iterations of (6)), but with the same memory footprint as the original method [12] (i.e. the old embeddings and the new adjacency matrix).

B. Real data

Finally, let us showcase our method in real-world graphs. To this end, we have used a football dataset which includes the yearly number of matches played between men's national teams [21]. We thus have a graph G_t per year, where nodes are national teams and an edge exists between two nodes if the corresponding teams have played a match between years $t-3$ and t and the weight is precisely the number of matches they have played during that period (we considered a moving four-year window so as to always include a world cup). Note that the number of edges vary, but the number of nodes too as national teams are formed (and not all national teams play a match every period).

Our algorithm is thus started at $t = 1930$ with the first world cup (and $n_t = 41$ nodes/national teams) using $d = 7$ as the embedding dimension, and we have proceeded to run the GD for every year until $t = 2016$ (with a graph size of $n_t = 222$).

As an illustrative example of the kind of tasks that our framework enables, let us consider the embedding corresponding to Australia. Along with New Zealand they formed the Oceania Football Confederation (OFC) in 1966, which they left to join the Asian Football Confederation (AFC) in 2006. This situation is clearly illustrated in figure 4, which shows the embeddings (a projection from the original 7-dimensional vectors for visualization) of Asian and Oceanian national teams for three years including the move between confederations. Note how until 2005 Australia is aligned with Oceanian teams' embeddings, until 2015 when it clearly becomes aligned with the Asian nodes' embeddings.

IV. Discussion and Future Work

We have presented a lightweight algorithm that is able to track the nodal positions of a stream of RDPG graphs G_t,

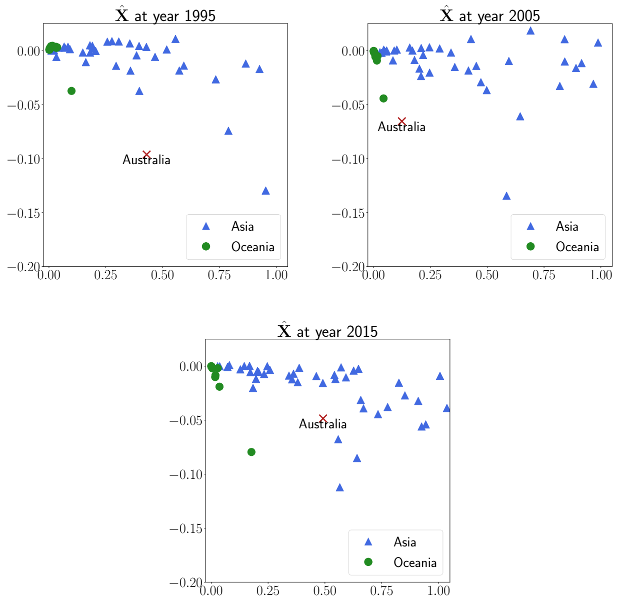

Fig. 4. A visualization of the embeddings' estimation of the proposed method for the graphs consisting of football matches between men's national teams. In particular, we focus on the nodal position of Australia, Asian and Oceanian teams for three years in particular. Note how the confederations are consistently aligned between years, and is Australia's embedding that changes from one community to the other, reflecting its confederation change in 2006.

efficiently avoiding the ever-present rotation problem of this kind of spectral embeddings (i.e. without recurring to re-alignments every time-step). This was achieved by reconsidering the underlying optimization problem in ASE and applying a convergent first-order gradient descent (GD) algorithm, which is warm-started from the embeddings' estimate of the previous time-step. Furthermore, the approach also enables inductive learning; i.e. embedding new nodes that are added to the network. This is achieved again by warm-starting the GD, in the case of the new nodes by projecting the corresponding new rows of the adjacency matrix to the space spanned by the previous embeddings. Finally, our code is freely available for experimentation at https://github.com/marfiori/efficient-ASE.

As future work, we plan on tackling the directed case. In this case, RDPG requires the estimation of two d-dimensional vectors, each corresponding to a direction. For instance, an edge from node i to j will exist with probability $\langle \mathbf{x}_i^l, \mathbf{x}_j^r \rangle$. In order to preserve its interpretability, it is necessary that the estimated embeddings matrices $\hat{\mathbf{X}}^l$ and $\hat{\mathbf{X}}^r$ are column-wise orthogonal [4], [14]. The addition of this constrain to the GD algorithm is a challenge that we are currently tackling. Another interesting theoretical development would be to establish properties of the resulting estimation of the GD, such as consistency, asymptotic normality or its stability to modifications on the embeddings.

References

[1] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart, "Representation learning for dynamic graphs: A survey," *Journal of Machine Learning Research*, vol. 21, no. 70, pp. 1–73, 2020.

[2] Yu Xie, Chunyi Li, Bin Yu, Chen Zhang, and Zhouhua Tang, "A survey on dynamic network embedding," *CoRR*, vol. abs/2006.08093, 2020.

[3] Guotong Xue, Ming Zhong, Jianxin Li, Jia Chen, Chengshuai Zhai, and Ruochen Kong, "Dynamic network embedding survey," *Neurocomputing*, vol. 472, pp. 212–223, 2022.

[4] A. Athreya, D. E. Fishkind, M. Tang, C. E. Priebe, Y. Park, J. T. Vogelstein, K. Levin, V. Lyzinski, and Y. Qin, "Statistical inference on random dot product graphs: A survey," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 8393–8484, Jan. 2017.

[5] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *Journal of Machine Learning Research*, vol. 23, no. 89, pp. 1–64, 2022.

[6] William L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[7] K. Levin, A. Athreya, M. Tang, V. Lyzinski, and C. E. Priebe, "A central limit theorem for an omnibus embedding of multiple random dot product graphs," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 964–967.

[8] A. Jones and P. Rubin-Delanchy, "The multilayer random dot product graph," *arXiv preprint arXiv:2007.10455*, 2020.

[9] I. Gallagher, A. Jones, and P. Rubin-Delanchy, "Spectral embedding for dynamic networks with stability guarantees," *Advances in Neural Information Processing Systems 34, NeurIPS*, 2021.

[10] Matthew Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and its Applications*, vol. 415, no. 1, pp. 20–30, 2006, Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[11] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu, "Timers: Error-bounded svd restart on dynamic networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[12] Keith Levin, Fred Roosta, Michael Mahoney, and Carey Priebe, "Out-of-sample extension of graph adjacency spectral embedding," in *Proceedings of the 35th International Conference on Machine Learning*. 10–15 Jul 2018, vol. 80, pp. 2975–2984, PMLR.

[13] P. D. Hoff, A. E. Raftery, and M. S. Handcock, "Latent space approaches to social network analysis," *J. Am. Stat. Assoc.*, vol. 97, no. 460, pp. 1090–1098, 2002.

[14] Bernardo Marenco, Paola Bermolen, Marcelo Fiori, Federico Larroca, and Gonzalo Mateos, "Online change point detection for weighted and directed random dot product graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 144–159, 2022.

[15] E.R. Scheinerman and K. Tucker, "Modeling graphs using dot product representations," *Comput. Stat*, vol. 25, pp. 1–16, 2010.

[16] P. Rubin-Delanchy, J. Cape, M. Tang, and C. E. Priebe, "A statistical interpretation of spectral embedding: The generalised random dot product graph," *arXiv:1709.05506 [stat.ML]*, 2017.

[17] Marcelo Fiori, Bernardo Marenco, Federico Larroca, Paola Bermolen, and Gonzalo Mateos, "Algorithmic advances for the adjacency spectral embedding," in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022, pp. 672–676.

[18] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi, "Dropping convexity for faster semi-definite optimization," in *Conference on Learning Theory*. PMLR, 2016, pp. 530–582.

[19] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht, "Low-rank solutions of linear matrix equations via procrustes flow," in *International Conference on Machine Learning*. PMLR, 2016.

[20] Y. Chi, Y. M Lu, and Y. Chen, "Nonconvex optimization meets low-rank matrix factorization: An overview," *IEEE Trans. on Signal Processing*, vol. 67, no. 20, pp. 5239–5269, 2019.

[21] Yang Li and Gonzalo Mateos, "Networks of international football: communities, evolution and globalization of the game," *Applied Network Science*, vol. 7, 2022.